Working Paper 5.1
April 24, 2008

AGEPRO Version 2.02 User's Guide

Jon K. T. Brodziak and Paul J. Rago
National Marine Fisheries Service
Northeast Fisheries Science Center
Population Dynamics Branch
166 Water Street
Woods Hole, Massachusetts 02543-1097
Email: Jon.Brodziak@NOAA.GOV

Version 2.02
23 July 2002

Abstract

This USER's GUIDE describes the AGEPRO model and version 2.02 software application for UNIX. At present there is no USER's GUIDE for the recently-developed PC version 2.02. However, both executables are compiled using the same FORTRAN 90 source code (see Appendix 2). Comparison tests show that the UNIX and PC versions 2.02 produce nearly identical outputs.

The AGEPRO model performs stochastic projections for an exploited age-structured fish population over a USER-specified time horizon. The model is designed to provide quantitative answers to "What if" questions about the effects of a harvest strategy on the age-structured population. The USER's GUIDE describes the numerical algorithms and theoretical basis of the model. Program inputs, outputs, structure and use are also described. Several examples of projection runs are used to illustrate general features of the model and complete Fortran 90 source code is provided.

Table of Contents

Introduction

The AGEPRO program performs stochastic projections of the abundance of an exploited age-structured population over a time horizon of up to 25 years. The primary purpose of the AGEPRO model is to characterize the sampling distribution of key fishery system outputs such as landings, spawning stock biomass, and recruitment under uncertainty. The acronym "AGEPRO" indicates that the program performs **age**-structured **pro**jections in contrast to size- or biomass-based projection models. In this program, the USER chooses the level of harvest that will be taken from the population by setting quotas or fishing mortality rates in each year of the time horizon.

There are three elements of uncertainty that can be incorporated in the AGEPRO model: **recruitment**, **initial population size**, and **natural mortality**. Recruitment is the primary stochastic element in the population model in AGEPRO, where recruitment is either the number of age-1 or age-2 fish in the population at the beginning of each year in the time horizon. There are a total of fifteen stochastic recruitment submodels that can be used for population projection. It should be noted that it is possible to simulate the case of deterministic recruitment with AGEPRO through a suitable choice of recruitment submodel and input data. Initial population size is a second potential source of uncertainty in AGEPRO that can be incorporated into population projection. To use this feature, the USER must have an initial distribution of population sizes that can be projected through the time horizon. Alternatively, the USER can choose to base the projections on a single estimate of initial population size. A third potential source of uncertainty in the AGEPRO model is natural mortality. In particular, the instantaneous natural mortality rate is assumed to be equal for all age classes in the population. The USER can choose to have a constant or a stochastic natural mortality rate. In the stochastic case, the natural mortality rates are taken to be realizations from a uniform distribution specified by the USER.

The AGEPRO model was conceived as part of a study to determine optimal strategies to rebuild a depleted fish stock. The AGEPRO model was initially developed in winter 1994 to compare the effects of various harvesting scenarios on a depleted stock. Subsequently, a manuscript describing the model was presented at the May 1994 meeting of the NEFSC Methods Working Group (Brodziak and Rago, manuscript1994; Brodziak et al. 1998). This software was then applied to assessment results for several stocks at the 18th SARC (NEFSC 1994) to evaluate the potential consequences of harvest policies. The model was extended in autumn 1994 to assist the Groundfish Plan Development Team and was also revised during summer 1995 to assist in the evaluation of Amendment 7 to the Northeast Multispecies Fishery Management Plan. Throughout these developments, the AGEPRO software was considered to be research software that had no documentation, except for comments in the source code. As a result, this USER'S GUIDE was written to provide documentation for the AGEPRO model and software. It was revised in July 1999 and again in February 2002 to describe modifications to the model and software. The most recent software is written in Fortran 90 (agepro_v2.f90).

Age-structured Population Model

An age-structured population model is the basis for the AGEPRO model and software. The age-structured model represents an iteroparous fish population whose abundance changes due to recruitment, natural mortality, and fishing mortality. Population size at age changes continuously throughout the year due to natural and fishing mortality which occur concurrently. That is, the fishery is "Type 2" Ricker (1975). Recruitment to the population is measured at the beginning of each year. Notation for the population model is summarized in Table 1.

Population Abundance, Survival, and Spawning Biomass

The AGEPRO model accounts for the number of fish alive within each ageclass of the population. The youngest ageclass comprises the recruits where the age of recruitment is either age-1 or age-2. The oldest ageclass comprises all fish that are alive and at least as old as a USER-specified cutoff age called the plus-group age. In this implementation the maximum number of ageclasses is 25. Let "R" denote the recruitment age and "A" denote the plus-group age. For each ageclass, the number of fish alive at the beginning of a given calender year (January 1st) is denoted as $N_j(t)$ where "j" is the ageclass and "t" is the year. In particular, note that $N_A(t)$ is the number of fish that are age-A or older at the beginning of year t. When the recruitment age is age, the population abundance at the beginning of year t can be succinctly represented as the vector **N**(t), where

$$\boldsymbol{N}(t) = \begin{bmatrix} N_1 \\ (,t,) \\ N_2 \\ (,t,) \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ N_A \\ (,t,) \end{bmatrix}$$

When the age of recruitment is age-2, the age classes are 2 through the plus-group and the age-1 element is not present.

Population survival at age from year t-1 to year t is calculated in a standard manner using instantaneous fishing and mortality rates. To describe annual survival through

mortality, let M(t) denote the instantaneous natural mortality rate and let $F_j(t)$ denote the instantaneous fishing mortality rate for age-j fish in year t. Then population size at age in

$$N_a(t) = N_{a-1}(t-1) \times e^{-M(t-1)-F_{a-1}(t-1)} \quad for\ a=R+1 \ \ A-1$$

$$N_A(t) = N_A(t-1) \times e^{-M(t-1)-F_A(t-1)} + N_{A-1}(t-1) \times e^{-M(t-1)-F_{A-1}(t-1)}$$

year t is given by
Note that survival for the plus-group involves an age-A and an age-(A-1) component. Also note that recruitment is determined through a stochastic process that is either dependent or independent of spawning biomass in year t-R (see **Stock-Recruitment Relationship** below).

Annual spawning biomass of the population is calculated from the population size vector **N**(t) and mortality rates as well as additional information concerning fish maturity and size at age. To describe annual mortality, let M(t) denote the instantaneous natural mortality rate and let $F_j(t)$ denote the instantaneous fishing mortality rate for age-j fish in year t. Let FM_j denote the average fraction of age-j fish that are mature and let $W_{S,j}$ denote the average spawning weight of an age-j fish. Further, let ZPROJ(t) denote the average fraction of total annual mortality that occurs from January 1st to the mid-point of the spawning season for the fish stock. Given these data, the vector of population size at the

$$\boldsymbol{N_S}(t) = \begin{bmatrix} N_1 \\ (,t,) \\ e^{-ZPROJ(t)[M(t)+F_1(t)]} \\ N_2 \\ (,t,) \\ e^{-ZPROJ(t)[M(t)+F_2(t)]} \\ . \\ . \\ . \\ . \\ . \\ . \\ N_A \\ (,t,) \\ e^{-ZPROJ(t)[M(t)+F_A(t)]} \end{bmatrix}$$

midpoint of the spawning season in year t, denoted by $\mathbf{N}_S(t)$, is

This vector is the result of applying instantaneous natural and fishing mortality rates that

occur prior to the spawning season to the population vector at the beginning of the year, **N**(t). When recruitment occurs at age-2, the age-1 element in $\mathbf{N}_S(t)$ is not present. The total spawning biomass in year t, denoted as SSB(t), is the sum over all ageclasses of the weight of mature fish at the midpoint of the spawning season,

$$SSB(t) = \sum_{a=1}^{A} W_{S,a} FM_a N_a(t) e^{-ZPROJ(t)[M(t)+F_a(t)]}$$

Catch, Landings, and Discards

In the AGEPRO model, the fraction of the population that is subject to harvest by the fishery depends on the fishery selectivity, also known as the partial recruitment pattern. The catch by ageclass in the fishery is given by the standard catch equation (Ricker 1975),

$$C_a(t) = \frac{F_a(t)}{M(t)+F_a(t)} \left\{ 1 - e^{-[M(t)+F_a(t)]} \right\} N_a(t)$$

and the catch of age-a fish in year t, denoted by $C_a(t)$, is

To account for age-specific discarding of fish, let $DF_a(t)$ be the fraction of age-a fish that are discarded and die in year t, and let $W_{L,a}$ and $W_{D,a}$ be the average weight at age-a for landed and discarded fish, respectively. The total landed weight in year t, denoted

$$L(t) = \sum_{a=1}^{A} C_a(t) \left[1 - DF_a(t) \right] W_{L,a}$$

by L(t), is

$$D(t) = \sum_{a=1}^{A} C_a(t) DF_a(t) W_{D,a}$$

Similarly, the total weight discarded in year t, denoted by D(t), is

Population Harvest

The AGEPRO model has two options for determining the level of population harvest in each year of the time horizon. The first option is a USER-specified fishing mortality rate (F-based management) and the second is a USER-specified landings quota (quota-based management). These harvest options can be mixed in a given projection run

where F-based management is applied in some years and quota-based management in the other years. The mix of quotas and F-based harvest can be useful when projecting forward from an old assessment where the catch is known in the intervening years.

When F-based management is applied, population harvest is determined by setting $F_a(t)$ in equation (4). In this case, the age-specific fishing mortality rate for age-a fish is the product of the fully-recruited fishing mortality rate, denoted by F(t), and the partial

$$F_a(t) = F(t)\,PR_a$$

recruitment to F(t) for age-a fish (age-specific selectivity), denoted by $PR_a(t)$. That is,

In equation (7), the USER specifies both F(t) and $PR_a(t)$. Based on $F_a(t)$, the landings and discards (if applicable), are determined by equations (5) and (6). When quota-based management is applied, however, the F(t) that would yield the landed quota must be determined numerically.

In the case of quota-based management, the target quota of landings in year t, denoted by Q(t), will translate into a variety of effective fishing mortality rates depending on population size, partial recruitment pattern, and discard pattern (if applicable). Ignoring the time component for a moment, the quota Q can be expressed as a function of F. That is Q= L(F), where F is the fully-recruited F and L is the level of landings as a function of F. To see this result, observe that equations (5) and (8) can be combined to give the catch of

$$C_a(F) = \frac{PR_a F}{M(t)+PR_a F}\left\{1-e^{-[M(t)+PR_a F]}\right\}N_a(t)$$

age-a fish as a function of F

Similarly, since the average weight of the landed fish and the fraction discarded do not

$$L(F) = \sum_{a=1}^{A} C_a(F)\left[1-DF_a\right]W_{L,a}$$

depend on F, landings can be expressed as a function of F

The fully-recruited fishing mortality, F, which satisfies the equation Q=L(F) can be found numerically using Newton's method. Details on this approach are found in **Appendix 1** Quotas which exceed the fishable biomass of the population are infeasible; conditions for defining infeasible quotas are also given in **Appendix 1.**

Stock-Recruitment Relationship

In general, stock and recruitment data are highly variable owing to intrinsic variability in factors governing survival and measurement error in the estimates of recruitment ($N_R(t)$) and spawning stock biomass that generated it (SSB(t-R)). It is usually desirable to characterize the uncertainty in recruitment predictions, but standard statistical approaches fail to provide the type of information desirable for such characterization. In the AGEPRO model, the stock-recruitment relationship ultimately defines the sustainable level of harvest and its expected variability over time. This follows from the model assumption that the stochastic processes of growth, maturation, and natural mortality are density-independent and stationary throughout the time horizon. Note that the stock-recruitment relationship does not affect the initial population abundance at the beginning of the time horizon (see **Initial Population Abundance**).

The AGEPRO model provides a total of fifteen stochastic recruitment models for population projection. Ten of the recruitment models are functionally dependent on SSB while five do not depend on SSB. The USER is responsible for the choice and parameterization of the recruitment model. The age of recruitment to the fish population is denoted as "R" and is either age-1 or age-2 as determined by the USER. A description of each of the recruitment models follows.

Model 1. Markov Matrix

A Markov matrix approach to modeling recruitment explicitly deals with uncertainty in the functional relationship and uncertainty in the estimation of population status. The Markov matrix contains transition probabilities that define the probability of obtaining a given interval range of recruitment given that SSB was within a defined interval range. In particular, the distribution of recruitment is assumed to follow a conditional multinomial distribution where the conditioning is with respect to appropriately lagged SSB.

Conventional methods for estimation of Markov matrices require far more data than are generally available in stock assessments. Matrix elements could be approximated by counting the number of times that a particular recruitment interval obtains from given SSB interval. In typical age-structured assessment with about 20 recruitment observations, one might be able to reliably define transition probabilities for a 2x2 matrix. Application of a Markov model exacerbates rather than solves, the model identifiability problem for stock and recruitment. For this reason, we describe a general bootstrapping approach that can be used to approximate the Markov matrix elements. In contrast to the usual Markov matrix approach, our estimates of transition probabilities are derived from variations about a hypothesized functional relationship between stock and recruitment, denoted by F(SSB, $\boldsymbol{\theta}$), where SSB is the appropriately lagged level of spawning stock biomass and $\boldsymbol{\theta}$ is a vector of parameters for the stock-recruitment relationship. In particular, recruitment is generated by

$$N_R(t) = F(SSB(t\text{-}R), \boldsymbol{\theta})$$

In general, nonlinear regression can be applied to obtain a point estimate of $\boldsymbol{\theta}$, denoted by $\boldsymbol{\theta}^*$, in Equation 10. Given the point estimate $\boldsymbol{\theta}^*$, bootstrap resampling of the residuals of the stock-recruitment model can be applied to estimate a set of **θ's**. In particular, suppose that there are T pairs of stock-recruitment values and let e_t denote the t^{th}

$$e_t = N_R(t) - F(SSB(t\text{-}R), \boldsymbol{\theta}^*)$$

residual from the model, where

Given the set of residuals (E), one bootstrap replicate ($D^{(b)}$) that consists of T pairs of $N_R(t)$ and SSB(t-R) can be generated by randomly selecting a set of T residuals with replacement from E. Let $E^{(b)}$ denote the random set of residuals, where $E^{(b)}=\{e_1^{(b)}, e_2^{(b)}, ..., e_T^{(b)}\}$ and let $Y(t)=F(SSB(t), \boldsymbol{\theta}^*)$ denote the t^{th} predicted recruitment. These residuals are then added to the set of original predicted recruitment values and a new parameter vector ($\boldsymbol{\theta}^{(b)}$) is computed based on the resulting bootstrapped stock-recruitment data set. In particular, the bootstrap replicate $D^{(b)}$ consists of T pairs of randomly generated recruitments and observed spawning stock biomasses where

The bootstrapped parameter vector $\boldsymbol{\theta}^{(b)}$ generated from $D^{(b)}$ defines one stock-recruitment curve. By creating a large number of bootstrap replicates and bootstrapped parameter vectors, a family of stock-recruitment curves can be generated. This family of curves can then be used to determine a Markov matrix as follows.

First, partition the spawning stock biomass axis (e.g. the horizontal axis) into J disjoint intervals and the recruitment axis (the vertical axis) into K disjoint intervals. The derived intervals can be defined as

$$I_j = [SSB_j, SSB_{j+1}]$$

$$O_k = [R_k, R_{k+1}]$$

where SSB_j and R_k denote endpoints of the disjoint intervals.

$$p_{j,k} = Prob\{N_R \mid O_k * SSB \mid I_j\}$$

The element in the j^{th} row and k^{th} column of the Markov matrix ($p_{j,k}$) is then

This probability can be approximated by the computing the number of points from the b^{th} bootstrapped stock recruitment curve that fall within the I_j x O_k cell and summing

those counts over all bootstrap replications. The number of points in a cell for a given stock-recruitment curve is controlled by the step size δ chosen for the SSB axis. For the range SSB_{min} to SSB_{max}, the number of points evaluated will be given as

$$H = \frac{SSB_{\max} - SSB_{\min}}{\delta}$$

If $C_{jk}^{(b)}$ represents the number of points in cell I_j x O_k for the b^{th} bootstrap realization, then the element in the j^{th} row and k^{th} column of the Markov matrix can be estimated as

$$Prob\{ N_R \mid O_k * SSB \mid I_j \} = \frac{\sum_{b=1}^{B} C_{jk}^{(b)}}{\sum_{k=1}^{K} \sum_{b=1}^{B} C_{jk}^{(b)}}$$

Thus for any set of J possible bootstrap realizations, there would be HJ data points computed.

Regardless of whether the method described above is used to determine a transition matrix, the AGEPRO software will generate stochastic recruitments based on a USER-specified Markov matrix when the recruitment flag is set to model 1. The USER can choose to have up to 25 recruitment levels and up to 10 SSB states (intervals). Here the expected recruitment levels ($N_{R,k}$) are the midpoints of the recruitment intervals O_k, that is, $N_{R,k} = (R_k + R_{k+1})/2$. For each SSB interval, the USER provides the conditional probability of realizing the expected recruitment level.

Model 2. Empirical Recruits Per Spawning Biomass Distribution

In some cases, it may be appropriate to assume that the distribution of recruits per spawner is independent of the number of spawners. The recruitment per spawning biomass (R/SSB) model randomly generates recruitment under the assumption that the distribution of the R/SSB ratio is stationary and independent of stock size. To describe this nonparametric approach, let Y_t be the R/SSB ratio for the t^{th} stock-recruitment data point

$$Y_t = \frac{N_R(t)}{SSB(t\text{-}R)}$$

and let R_s represent the s^{th} element in the ordered set of Y_t. The probability density function for R_s, denoted as $f(R_s)$, is 1/T for all values of R/SSB $\in$ R_s where T = the number of stock-recruitment data points. Let $F(R_s)$ denote the cumulative distribution function

$$F(R_S) = \frac{s\text{-}1}{T\text{-}1}$$

(cdf). Let $F(R_{min}) = 0$ and $F(R_{max}) = 1$ so that the cdf of R_s can be written as

Random values of R/SSB can be generated by applying the probability integral transform to the empirically derived cdf. Let U be a uniformly distributed random variable on the interval [0,1]. The value of R/SSB corresponding to U is determined by applying the inverse of the cdf $F(R_s)$. In particular, when U is an integer multiple of 1/(T-1) so that U=s/(T-1) then R/SSB = $F^{-1}(U) = R_s$. Otherwise R/SSB can be obtained by

$$U = \left(\frac{\frac{s}{T\text{-}1} - \frac{s\text{-}1}{T\text{-}1}}{R_{S+1} - R_S}\right)\left[R^aSSB - R_S\right] + \left(\frac{s\text{-}1}{T\text{-}1}\right)$$

linear interpolation. In particular, if (s-1)/(T-1) < U < s/(T-1), then

$$R^aSSB = (T\text{-}1)(R_{S+1} - R_S)\left(U - \frac{s\text{-}1}{T\text{-}1}\right) + R_S$$

Solving for R/SSB as a function of U yields

where the interpolation index s (Equation 19) is determined as the greatest integer in

$$N_R(t) = SSB(t\text{-}R)\,C\,R^aSSB$$

1+U(T-1). Given a random value of R/SSB, recruitment is then generated as
The AGEPRO software can generate stochastic recruitments based on a USER-specified set of stock-recruitment data. There can be up to 100 stock-recruitment data points.

Model 3. Empirical Recruitment Distribution

An empirical model for the estimation of recruitment is to draw randomly from the observed set of recruitments { $N_R(1)$, $N_R(2)$, ..., $N_R(T)$} taken from the assessment. Here the recruitment distribution is a multinomial random variable where the probability of randomly choosing a particular recruitment level is 1/T given T observed recruitments. This approach is nonparametric and assumes that future recruitment is totally independent

of spawning stock biomass. When current levels of SSB are near the midrange of historical values this assumption is acceptable. However, if contemporary SSB values are near the bottom of the range, then this approach could be dangerously optimistic, for it assumes that all historically observed recruitment levels are possible, irrespective of SSB. The AGEPRO software permits up to 100 observed recruitments for the USER-specified recruitment distribution.
Note that the empirical recruitment distribution model can be used to make deterministic projections by specifying a single observed recruitment. In this case, recruitment will be constant throughout the time horizon.

Model 4. Two-Stage Empirical Recruits Per Spawning Biomass Distribution

The two-stage recruits per spawning biomass model is a direct generalization of the R/SSB model where the spawning stock of the population is categorized into "low" and "high" states. In particular, there is an R/SSB distribution for the low SSB state and an R/SSB distribution for the high SSB state. Let F_{LOW} be the cdf and let T_{LOW} be the number of R/SSB values for the low SSB state. Similarly, let F_{HIGH} be the cdf and let T_{HIGH} be the number of R/SSB values for the high SSB state. Further, let SSB* denote the cutoff level of SSB such that, if SSB$\geq$SSB*, then SSB is considered to be in the high state, while if SSB<SSB* then SSB is in the low state. Recruitment is stochastically generated from F_{LOW} and T_{LOW} using Equations (20) and (21) when SSB is in the low state. Conversely, Recruitment is stochastically generated from F_{HIGH} and T_{HIGH} using Equations (20) and (21) when SSB is in the high state. The AGEPRO software can generate stochastic recruitments for the two-stage model given a USER-specified set of stock-recruitment data of up to 100 stock-recruitment data points per SSB state.

Model 5. Beverton-Holt Curve With Lognormal Error

The Beverton-Holt curve with lognormal errors is a parametric model of recruitment generation where survival to recruitment age is density dependent and subject to stochastic variation. The Beverton-Holt curve with lognormal error generates

$$n_R(t) = \frac{a \cdot ssb(t\text{-}R)}{b + ssb(t\text{-}R)} \cdot e^{w}$$

$$\text{where } w \sim N(0, \sigma_w^2),$$
$$N_R(t) = c_R \cdot n_R(t),$$
$$SSB(t\text{-}R) = c_{SSB} \cdot ssb(t\text{-}R)$$

recruitment as
where the stock-recruitment parameters "a", "b", and "σ_w^2" and the conversion coefficients for recruitment (c_R) and spawning stock biomass (c_{SSB}) are specified by the USER. Here it

is assumed that the parameter estimates for the Beverton-Holt curve have been estimated in relative units determined by the USER (e.g. $n_R(t)$ and ssb(t-R)) and then converted to absolute values with the conversion coefficients. *In the AGEPRO software, the absolute value for recruitment is numbers of fish, while for SSB, the absolute value is kilograms of SSB.* For example, if the stock-recruitment curve was estimated with stock-recruitment data that were measured in millions of fish and thousands of metric tons of SSB, then $c_R=10^6$ and $c_{SSB}=10^6$. Note that it is common to estimate the parameters of the stock-recruitment curve in relative units to reduce the potential effects of roundoff error on parameter estimates.

Model 6. Ricker Curve With Lognormal Error

The Ricker curve with lognormal error is a parametric model of recruitment generation where survival to recruitment age is density dependent and subject to stochastic

$$n_R(t) = a \cdot ssb(t\text{-}R) \cdot e^{-b \cdot ssb(t\text{-}R)} \cdot e^{w}$$

$$\begin{aligned} \text{where } w &\sim N(0, \sigma_w^2), \\ N_R(t) &= c_R \cdot n_R(t), \\ SSB(t\text{-}R) &= c_{SSB} \cdot ssb(t\text{-}R) \end{aligned}$$

variation. The Ricker curve with lognormal error generates recruitment as where the stock-recruitment parameters "a", "b", and "σ_w^2" and the conversion coefficients for recruitment (c_R) and spawning stock biomass (c_{SSB}) are specified by the USER. It is assumed that the parameter estimates for the Ricker curve have been estimated in relative units determined by the USER (e.g. $n_R(t)$ and ssb(t-R)) and then converted to absolute values with the conversion coefficients. Note that the absolute value for recruitment is numbers of fish, while for SSB, the absolute value is kilograms of SSB.

Model 7. Shepherd Curve With Lognormal Error

The Shepherd curve with lognormal error is a parametric model of recruitment generation where survival to recruitment age is density dependent and subject to stochastic

$$n_R(t) = \frac{a \cdot ssb(t\text{-}R)}{1+\left(\frac{ssb(t\text{-}R)}{k}\right)^b} \cdot e^{w}$$

$$\begin{aligned} \text{where } w &\sim N(0, \sigma_w^2), \\ N_R(t) &= c_R \cdot n_R(t), \\ SSB(t\text{-}R) &= c_{SSB} \cdot ssb(t\text{-}R) \end{aligned}$$

variation. The Shepherd curve with lognormal error generates recruitment as where the stock-recruitment parameters "a", "b", "k", and "σ_w^2" and the conversion coefficients for recruitment (c_R) and spawning stock biomass (c_{SSB}) are specified by the USER. It is assumed that the parameter estimates for the Shepherd curve have been estimated in relative units determined by the USER (e.g. $n_R(t)$ and ssb(t-R)) and then converted to absolute values with the conversion coefficients. Note that the absolute value for recruitment is numbers of fish, while for SSB, the absolute value is kilograms of SSB.

Model 8. Lognormal Distribution

The lognormal distribution provides a parametric model for stochastic recruitment

$$n_R(t) = e^w$$

$$where\ e^w \sim LN(\mu_{logR}, \sigma^2_{logR})$$
$$N_R(t) = c_R \mathrm{C} n_R(t)$$

generation that is independent of spawning stock biomass. The lognormal distribution is where the lognormal distribution parameters μ_{logR} and σ_{logR} (USER inputs the standard deviation of logR, not the variance) and the conversion coefficients for recruitment (c_R) and spawning stock biomass (c_{SSB}) are specified by the USER. It is assumed that the parameters of the lognormal distribution have been estimated in relative units determined by the USER (e.g. $n_R(t)$) and then converted to absolute values with the conversion coefficients. Note that the absolute value for recruitment is numbers of fish, while for SSB, the absolute value is kilograms of SSB.

Model 9. Time-Varying Empirical Recruitment Distribution

The time-varying empirical recruitment distribution model is an extension of model 3. This empirical model for the estimation of recruitment draws randomly from a set of T recruitments levels for year t of the time horizon { $N_R(t,1)$, $N_R(t,2)$, ..., $N_R(t,T)$ }. Here the recruitment distribution for each year of the time horizon is a multinomial random variable where the probability of randomly choosing a particular recruitment level is 1/T given T levels of recruitment. This approach is nonparametric and assumes that future recruitment is totally independent of spawning stock biomass. Further, it is the responsibility of the USER to determine an appropriate set of recruitment levels for each year of the time horizon. The AGEPRO software permits up to 100 observed recruitments for the USER-specified recruitment distribution for each year of the time horizon. The USER must input T potential recruitment levels in each year for a total of TY recruitment inputs.

As with model 3, the time-varying empirical recruitment distribution model can be used to make deterministic projections by specifying a single recruitment level for each

year of the time horizon. In this case, recruitment will be a USER-defined constant in each year of the time horizon.

Model 10. Beverton-Holt Curve With Autocorrelated Lognormal Error

The Beverton-Holt curve with autocorrelated lognormal errors is a parametric model of recruitment generation where survival to recruitment age is density dependent and subject to serially-correlated stochastic variation. The Beverton-Holt curve with

$$n_R(t) = \frac{a \cdot ssb(t-R)}{b+ssb(t-R)} \cdot e^{\varepsilon}$$

$$\text{where } \varepsilon_t=\varepsilon_{t-1}\varphi+w_t \quad w \sim N(0,\sigma_w^2),$$
$$N_R(t) = c_R \cdot nsubR(t),$$
$$SSB(t-R) = c_{SSB} \cdot ssb(t-R)$$

lognormal error generates recruitment as
where the stock-recruitment parameters “a”, “b”, and “σ_w^2", the serial correlation parameter “φ”, and the conversion coefficients for recruitment (c_R) and spawning stock biomass (c_{SSB}) are specified by the USER.

Model 11. Ricker Curve With Autocorrelated Lognormal Error

The Ricker curve with autocorrelated lognormal error is a parametric model of recruitment generation where survival to recruitment age is density dependent and subject to serially-correlated stochastic variation. The Ricker curve with autocorrelated lognormal

$$n_R(t) = a \cdot ssb(t-R) \cdot e^{-b \cdot ssb(t-R)} \cdot e^{\varepsilon}$$

$$\text{where } \varepsilon_t=\varepsilon_{t-1}\varphi+w_t \quad w \sim N(0,\sigma_w^2),$$
$$N_R(t) = c_R \cdot nsubR(t),$$
$$SSB(t-R) = c_{SSB} \cdot ssb(t-R)$$

error generates recruitment as
where the stock-recruitment parameters “a”, “b”, and “σ_w^2", the serial correlation parameter “φ”, and the conversion coefficients for recruitment (c_R) and spawning stock biomass (c_{SSB}) are specified by the USER.

Model 12. Shepherd Curve With Autocorrelated Lognormal Error

The Shepherd curve with autocorrelated lognormal error is a parametric model of recruitment generation where survival to recruitment age is density dependent and subject to serially-correlated stochastic variation. The Shepherd curve with autocorrelated lognormal error generates recruitment as

$$n_R(t) = \frac{a \cdot ssb(t-R)}{1+\left(\frac{ssb(t-R)}{k}\right)^b} \cdot e^{\varepsilon}$$

$$\text{where } \varepsilon_t = \varepsilon_{t-1}\varphi + w_t \quad w \sim N(0, \sigma_w^2),$$
$$N_R(t) = c_R \cdot n_R(t),$$
$$SSB(t-R) = c_{SSB} \cdot ssb(t-R)$$

where the stock-recruitment parameters “a”, “b”, “k”, and “σ_w^2", the serial correlation parameter “φ”, and the conversion coefficients for recruitment (c_R) and spawning stock biomass (c_{SSB}) are specified by the USER.

Model 13. Autocorrelated Lognormal Distribution

The autocorrelated lognormal distribution provides a parametric model for stochastic recruitment generation with serial correlation that is independent of spawning

$$n_R(t) = e^{\varepsilon}$$

$$\text{where } \varepsilon_t = \varepsilon_{t-1}\varphi + w_t \quad w \sim N(\mu_{logR}, \sigma_{logR}^2)$$
$$N_R(t) = c_R \cdot n_R(t)$$

stock biomass. The autocorrelated lognormal distribution is
where the lognormal distribution parameters μ_{logR} and σ_{logR} (USER inputs the standard deviation of logR, not the variance), the serial correlation parameter “φ”, and the conversion coefficients for recruitment (c_R) and spawning stock biomass (c_{SSB}) are specified by the USER.

Model 14. Empirical Cumulative Distribution Function of Recruitment

The empirical cumulative distribution function (cdf) of recruitment can be used to randomly generates recruitment under the assumption that the distribution of the R is

stationary and independent of stock size. To describe this nonparametric approach, let R_s represent the s^{th} element in the ordered set of observed recruitment values. The probability density function for R_s, denoted as $f(R_s)$, is 1/T for all values of $R \in R_s$ where T is the number of stock-recruitment data points. Let $F(R_s)$ denote the cumulative distribution

$$F(R_S) = \frac{s\text{-}1}{T\text{-}1}$$

function (cdf). Let $F(R_{min}) = 0$ and $F(R_{max}) = 1$ so that the cdf of R_s can be written as

Random values of R can be generated by applying the probability integral transform to the empirically derived cdf. Let U be a uniformly distributed random variable on the interval [0,1]. The value of R corresponding to U is determined by applying the inverse of the cdf $F(R_s)$. In particular, when U is an integer multiple of 1/(T-1) so that U=s/(T-1) then $R = F^{-1}(U) = R_s$. Otherwise R can be obtained by linear

$$U = \left(\frac{\frac{s}{T\text{-}1} - \frac{s\text{-}1}{T\text{-}1}}{R_{S+1} - R_S} \right) \left[R - R_S \right] + \left(\frac{s\text{-}1}{T\text{-}1} \right)$$

interpolation. In particular, if (s-1)/(T-1) < U < s/(T-1), then

$$R = (T\text{-}1)(R_{S+1} - R_S)\left(U - \frac{s\text{-}1}{T\text{-}1} \right) + R_S$$

Solving for R as a function of U yields

where the interpolation index s is determined as the greatest integer in 1+U(T-1). The AGEPRO software can generate stochastic recruitments based on a USER-specified set of up to 100 stock-recruitment data points.

Model 15. Two-Stage Empirical Cumulative Distribution Function of Recruitment

The two-stage empirical cumulative distribution function of recruitment model is a direct generalization of Model 14 where the spawning stock of the population is categorized into "low" and "high" states. In particular, there is a cdf for R when the population is in the low SSB state and a cdf for R when the population is in the high SSB state. Let F_{LOW} be the cdf and let T_{LOW} be the number of R values for the low SSB state. Similarly, let F_{HIGH} be the cdf and let T_{HIGH} be the number of R values for the high SSB state. Further, let SSB* denote the cutoff level of SSB such that, if SSB$\geq$SSB*, then SSB is considered to be in the high state, while if SSB<SSB* then SSB is in the low state. Recruitment is stochastically generated from F_{LOW} and T_{LOW} using Equations (32) and (33) when SSB is in the low state. Conversely, Recruitment is stochastically generated from F_{HIGH} and T_{HIGH} using Equations (32) and (33) when SSB is in the high state. The

AGEPRO software can generate stochastic recruitments for the two-stage model given a USER-specified set of stock-recruitment data of up to 100 stock-recruitment data points.

Constrained Recruits Per Spawning Biomass For Lognormal Error Models

The lognormal error terms for the six parametric recruitment models and the two lognormal distribution models can produce unusual realizations of R/SSB in a projection analysis because an estimated lognormal distribution can be highly skewed with a very wide tail. The impact of an unusual recruitment can be substantial in a projection analysis. Therefore, realized R/SSB values can be constrained in the AGEPRO model for the 4 stock-recruitment models that use the lognormal distribution. There are two potential constraints are based on the level of SSB within the stock. Let SSB_{CUT} denote a cutoff level of SSB, and let [L_{LOW} , U_{LOW}] and [L_{HIGH} , U_{HIGH}] denote two intervals. Whenever SSB(t) <SSB_{CUT}, then the realized R/SSB value generated from the recruitment model must be within the interval [L_{LOW} , U_{LOW}]. If the realized R/SSB falls outside this interval, additional recruitments are simulated with the stochastic recruitment model until one falls within the constraining interval. Similarly, whenever SSB(t) $\geq$$SSB_{CUT}$, then the realized R/SSB value generated from the recruitment model must be within the interval [L_{HIGH} , U_{HIGH}]. If R/SSB values are expected to be more variable when SSB is above SSB_{CUT}, then it is natural to choose to have the interval [L_{LOW} , U_{LOW}] to be within the interval [L_{HIGH} , U_{HIGH}]. In this case, the endpoints of the intervals are ordered as $L_{HIGH} < L_{LOW} < U_{LOW} < U_{HIGH}$.

The use of R/SSB constraints can be appropriate when the population is near an historic low level of SSB. In this case, it would be natural to consider SSB_{CUT} to be the historic minimum value of SSB. Extrapolation of what R/SSB values would result if SSB(t) falls below SSB_{CUT} could have substantial influence on determining a rebuilding strategy for the population. For example, one might constrain the realized R/SSB values when SSB(t) falls below SSB_{CUT} to be between the 10^{th} and 90^{th} percentiles of the empirical R/SSB distribution taken from the assessment. When SSB(t) is above SSB_{CUT}, one might consider other bounds on the R/SSB values such as 1/100 of the minimum observed R/SSB value or 100 times the maximum observed R/SSB value. Similar comments apply for a population that is near its historic maximum level of SSB. While the AGEPRO software requires the USER to specify a lower and an upper constraining interval for R/SSB values when the R/SSB constraint option is chosen, it should be noted that the USER can use a single interval by either (i) setting the intervals to be equal or (ii) setting SSB_{CUT} to be 0.

Initial Population Abundance

The initial population abundance, **N**(1), plays a key role in the AGEPRO model and is defined as the absolute number of fish alive on January 1^{st} of the first year of the time horizon. Note that **N**(1) is determined by the assessment model and is entirely independent of the stock-recruitment model used in the projections. When the recruitment age is age-1, then **N**(1) is sufficient to begin the projection. However, when the recruitment

$$\frac{N_A(0)}{N_{A\text{-}1}(0)} = \sum_{k=1}^{infinity} e^{-(F+M)k} = \gamma \qquad 19$$

age is age-2, additional information on fishing mortality at age in the prior year (year 0) is needed to project recruitment from population size in year 0. In particular, the vector **F**(0) of instantaneous fishing mortalities at age is needed, and the partial recruitment vector (**PR**(0)) in the prior year must also be provided. This information determines the abundances $N_2(0)$, $N_3(0)$, ..., $N_{A-2}(0)$. An additional approximation is needed to back calculate $N_{A-1}(0)$ and $N_A(0)$ because both agegroups are combined in $N_A(1)$. Assuming that recruitment is constant, that fully-recruited fishing mortality is constant, and that both agegroups are fully recruited, the ratio of $N_A(0)$ to $N_{A-1}(0)$ would be determined by the magnitude of fishing mortality as

for γ a constant. Therefore, based on the definition of survival to the plus-group (Equation 2), the back calculated values for $N_{A-1}(0)$ and $N_A(0)$ are

There are two possibilities for determining the initial population abundance in the AGEPRO model. The first approach is to use the point estimate of **N**(1) taken from the assessment along with F(0) **PR**(0) if age-2 recruitment is used. In this case, the best estimate of population abundance is being used for projection and this estimate is assumed to perfectly characterize the initial state of the population. In effect, there is no uncertainty in the initial state of the population and the purpose of the projection is to characterize the sampling distribution of key fishery outputs given the initial population size is perfectly known. The second approach is to provide a set of initial population vectors, { $\mathbf{N}^{(1)}(1)$,

$$N_{A-1}(0) = \frac{N_A(1) \, C \, e^{[F+M]}}{1+\gamma}$$

$$N_A(0) = \gamma \, C \, N_{A-1}(0)$$

$\mathbf{N}^{(2)}(1)$, ..., $\mathbf{N}^{(B)}(1)$ }, that represent the sampling distribution of the estimator of **N**(1). In this case, the assessment model is the estimator of **N**(1) and the purpose of the projection is to characterize the sampling distribution of key fishery outputs given the uncertainty in the estimate of the initial population size. The nonparametric bootstrap provides one method to generate a sampling distribution for the estimator of **N**(1). This method has been applied to the ADAPT tuned-VPA model to generate sampling distributions of **N**(1) in a variety of assessments. If the recruitment age is age-2, it is necessary to provide a set of fully-recruited fishing mortality rates in the previous year, { $F^{(1)}(0)$, $F^{(2)}(0)$, ..., $F^{(B)}(0)$ }, and **PR**(0), where **PR**(0) is constant.

Regardless of whether the initial population abundance is a single **N**(1) or a distribution of **N**(1)'s, the USER must also specify the units of the initial population size vector taken from the assessment model. In particular, it is assumed that the USER provides a scaled **n**(1) or a distribution of **n**(1)'s that are in relative units, and also provides the conversion coefficient (c_N) that converts **n**(1) to absolute numbers of fish where $\mathbf{N}(1) = c_N \cdot \mathbf{n}(1)$.

Stochastic Natural Mortality

Natural mortality is often assumed to be constant over recruited ageclasses and to be equal to its long-term average for assessment purposes. While this simplifying assumption is important for estimation of historic and current population sizes, it may be useful to consider the potential effects of variation in the instantaneous natural mortality rate when performing projections. To this end, it is possible to set natural mortality to be a random variable in the AGEPRO model. In particular, natural mortality can be modeled as a uniform random variable for simulation. That is, the natural mortality rate in year t would be distributed as $M(t) \sim UNIFORM[L_M , U_M]$, where L_M and U_M are USER-specified constants. Note that the realized natural mortality rate applies to all ageclasses within **N**(t).

Although it is likely difficult to estimate L_M and U_M directly, the USER can choose L_M and U_M so that the annual probability of survival in the absence of fishing varies uniformly as follows. The annual probability of survival (S) is related to the long-term instantaneous natural mortality rate (M) as $S = e^{-M}$. If it is reasonable that S varies uniformly between S-d and S+d for some constant d, then set $L_M = - \ln(S+d)$ and set $U_M = -\ln(S-d)$. For example, if M=0.30 and d=0.1, $L_M = 0.1734$ and $U_M = 0.4450$ would be the bounds for the uniform random variable. Note that in this example, the interval $[L_M , U_M]$ is not symmetric about M=0.30 although $S \sim UNIFORM[0.64 , 0.84]$. In the AGEPRO software, the USER specifies the interval $[L_M , U_M]$ when the stochastic natural mortality option is chosen.

Spawning Stock Biomass Threshold

Whether or not a given harvest policy will likely lead to a threshold level of spawning stock biomass is potentially important for evaluation of the merits of that harvest policy. In the AGEPRO software, the USER can specify a threshold level of SSB, denoted by $SSB_{THRESHOLD}$, for Sustainable Fisheries Act policy evaluation. This is called the SFA-threshold option. Let $K_{THRESHOLD}(t)$ be the number of times that SSB(t) meets or exceeds the $SSB_{THRESHOLD}$ in year t where $K_{THRESHOLD}(t)$ is computed for each year using the entire set of simulations. An estimate of the probability that $SSB_{THRESHOLD}$ would be met or

$$Prob\{ SSB(t) \$ SSB_{THRESHOLD} \} = \frac{K_{THRESHOLD}(t)}{K_{TOTAL}(t)}$$

exceeded in year t is then computed as

where $K_{TOTAL}(t)$ is the total number of feasible simulation runs of year t. Note that with quota-based management, it is possible for a harvest quota to be unattainable for a given year and simulation because the population abundance is too low to allow the quota to be taken. When this occurs, the simulation run is defined to be infeasible and the simulation run ceases. In this case, the denominator in the right-hand side of equation (36) would not

count that simulation run in the computation of the probability that $SSB_{THRESHOLD}$ would be met or exceeded.

Threshold values for total stock biomass (measured on January 1st), mean stock biomass (as defined by the average numbers at age vector and stock weights at age), and threshold fishing mortality are evaluated in a similar manner.

Target Fishing Mortality

In some cases, it may be informative to have the specified fishing mortality rate for F-based management change when the $SSB_{THRESHOLD}$ is met or exceeded. This might be reasonable, for example, if the $SSB_{THRESHOLD}$ represented a rebuilding target for SSB. In this case, the level of F might be increased from a low rebuilding level in the year after the target was achieved.

The AGEPRO software gives the USER the option to specify a target F, F_{TARGET} , that will be applied in the year following the year in which the $SSB_{THRESHOLD}$ is met or exceeded. This is called the F-target option. Note that this option requires that the SSB-threshold option be chosen. Also note that the F-target option is dynamic so that F_{TARGET} can be applied only in a year following the year in which the $SSB_{THRESHOLD}$ is met or exceeded.

In addition to specifying a target F, the USER must also specify a calendar year in the time horizon when the F-target option begins, denoted by $Y_{F\text{-target}}$. For example, if the time horizon were the interval of years [1998, 2007] , then $Y_{F\text{-target}}$ might be chosen to be 2005 if this was a policy option under consideration. Alternatively, choosing $Y_{F\text{-target}}$ to be any year less than the first year in the time horizon would mean that the F-target option would begin in the first year. This implies that F_{TARGET} could be applied in the second year of the time horizon.

Prediction of Catch-At-Age Index

In some cases, it may be useful to predict a catch-at-age index from a research survey based on the simulated population sizes from the AGEPRO model. A predicted survey index could, for example, provide a measure of yearclass strength which might be a consideration in the evaluation of a harvest policy if a certain index level is reached. The AGEPRO software gives the option of predicting a survey catch-at-age index based on a predictive model specified by the USER. Let p denote the age of the fish for the catch-at-age index and let s_p be the predicted value of the survey index. It is assumed that the predictive model is linear in the scaled numbers of fish at age-p. That is,

$$s_p = \beta_0 + \beta_1 C \frac{N_p(t)}{c_N}$$

where c_N is the conversion coefficient for the initial population size vector. Here the USER must determine the coefficients β_0 and β_1 prior to the projection analysis. Further, the USER must specify a critical value of the survey index, s_p*, that will be compared to the realized predictions. In particular, the probability that the critical survey index would likely be exceeded in each year of the time horizon is evaluated and output.

Landings by Market Category

In some cases, it may be useful to partition projected landings into market categories for further analysis. This might be done, for example, if there was a need to evaluate the expected benefits from a harvest policy through fishery revenues where fish price varied by market category. The AGEPRO software can partition landings at age into market categories and provide the number of fish and weight of fish in up to three USER-specified market categories. In particular, the market category option requires the USER to specify the proportion of each age class that constitute each market category. Let $q_{a,j}$ denote the proportion of age-a fish in the j^{th} market category and let MC denote the number of market categories. Then the total number of fish ($L_{N,j}$ (t)) and the total weight

$$L_{N,j} = \sum_{a=R}^{A} q_{a,j} C_a(t) C (1 - DF_a(t))$$

$$L_{W,j} = \sum_{a=R}^{A} q_{a,j} C_a(t) W_{L,a} C (1 - DF_a(t))$$

of fish ($W_{N,j}$(t)) in the j^{th} market category in year t are computed as
Note that the USER must specify the proportions $q_{a,j}$ for each ageclass in **N**.

Age-Structured Projection Software

The age-structured projection software (agepro_v2) was revised during the spring of 1996, the summer of 1999, and the winter of 2002, to incorporate the useful features of various versions of the program that were created for specific applications. As a result, USER-supplied input files for previous versions of the code will need revision to be compatible with the new program. The required adjustments to older input files, however, are minor in all cases.

This part of the USER'S GUIDE provides operational details for the AGEPRO software and is organized into four sections. First, *input data* requirements and projection options are covered. The structure of an input file is described and the use of a general program (SETPRO) to create a new input file or describe an existing input file for is discussed. Second, *model outputs* are described in relation to logical flags in the input file. The structure of an output file is also described. Third, a section on *program structure* identifies the flow of data and calculations within the AGEPRO software is provided. Fourth, a set of *examples* are provided to identify the flow of data and calculations within the AGEPRO software. Last, a listing of the AGEPRO source code is provided in **Appendix 2**.

Input Data

There are four categories of input data for an AGEPRO projection run: *system*, *simulation*, *biological*, and *fishery*. The *system* data are read from standard input (e.g. from a terminal or via input redirection) while the *simulation*, *biological* and *fishery* data are read from an input file. A description of each data category follows.

The *system* data are the file names for the input and output files for the projection run. These data are required to run the AGEPRO software. The USER can enter these file names either from the terminal or through input redirection (see below). With terminal input, the USER is first prompted for an input file name and then for an output file name. That is, at the UNIX prompt ">" the USER would enter

```
>agepro.exe
>Enter the input filename:
>my_input_filename
>
>Enter the output filename:
>my_output_filename
>
>Projection analysis is running ...
>
>Projection analysis has been completed.
```

The AGEPRO software checks whether the input file exists and prompts the USER for another filename if the input file does not exist. Similarly, the software checks whether the output file already exists and prompts the USER for another filename if the output file already exists.

The USER can also apply input redirection to specify the input and output file names in a file. For example, assume that the executable software is named "agepro.exe" and that the USER has created a file called "input" with two lines where the first line is the input filename and the second line is the output filename. To use input redirection to run AGEPRO, the USER would type

```
>agepro.exe  <  input
```

at the UNIX system prompt. **The use of input redirection is recommended for background processing.** In particular, the USER would type

```
>agepro.exe  <  input &
```

at the UNIX system prompt to run the projection in the background. Here it is worth noting that a large number of projection runs can be run sequentially in the background by writing a C Shell script loops over a set of input and output file names. It is not recommended that several projections be run at the same time on a single UNIX workstation due to the memory requirements of the AGEPRO software in a multitasking environment. Experience has shown that running several projections simultaneously can cause system crashes.

The *simulation* data are the inputs needed to setup and define the simulation run. These data are required to run the AGEPRO software and are read from the input file (Tables 2 and 3). The first input is a character string that describes the projection run. The string can be up to 30 characters and should have no blanks. The second input is the base year of the time horizon, which is the first year of the projection. The third input is the length of the time horizon, which can be up to 25 years. The fourth input is the number of simulations to perform for each initial population vector; there can be up to 200 simulations for each initial condition. The fifth input initializes the random number generator and in particular, gives the number of times to call the pseudo-random number generator "ran2" (Press et al. 1992) before using the random numbers. Note that this generator can generate a vast number (approximately $2 \cdot 10^{18}$) pseudo-random numbers before repeating itself. In comparison, a projection analysis that required 2 random numbers per simulated year (1 for randomization of recruitment and 1 for randomization of M) would need only $5 \cdot 10^{6}$ random numbers assuming 500 initial population sizes, 200 simulations per initial population size, and 25 years in the time horizon.

The sixth through the twentieth inputs are logical flags that determine simulation properties (Tables 2 and 3). The sixth input is the age-2 recruitment flag. If true, recruitment age is age-2; otherwise it is age-1. The seventh input is the mixture flag for

harvesting. If true, catch projections are based on a mixture of F-based and quota-based management by year; otherwise, the harvest is based on one management strategy. The eighth input is the discard flag. If true, discards at age are included in the projection analysis; otherwise, no discards are included in the analysis. The ninth input is the quota-based management flag. If true, catch projections are based on quotas; otherwise catch projections are F-based. The tenth input is the constant harvest strategy flag. If true, the harvest strategy does not change in time, e.g. the F or the quota is fixed; otherwise the harvest strategy can vary from year to year. The eleventh input is the F-target flag. If true, then a target value of F is applied in the year after any year when the SSB threshold is achieved; otherwise no change occurs. The twelfth input is the index flag. If true, a prediction of an age-specific recruitment index is made; otherwise no prediction is made. The thirteenth input is the SFA threshold flag. If true, realized SSB, total stock biomass, mean stock biomass, fully-recruited fishing mortality, and biomass-weighted fishing mortality are compared to a threshold level; otherwise no comparisons are made. The fourteenth input is the market category flag. If true, landings are summarized by market category and output to file; otherwise no market category summaries are made. The fifteenth input is the total mortality flag. If true, the fraction of total mortality that occurs prior to spawning can vary from year to year; otherwise there is no annual variation. The sixteenth input is the partial recruitment flag. If true, the partial recruitment to fishing mortality vector can vary from year to year; otherwise there is no annual variation.The seventeenth input is the constant discard flag. If true, the fraction discarded at age is constant; otherwise the fraction discarded at age can vary from year to year. The eighteenth input is the bounded recruitment flag. If true, then realized recruitments generated with the lognormal, Beverton-Holt, Ricker, and Shepherd stock-recruitment models will be bounded based on realized R/SSB ratios; otherwise no bounds are applied. The nineteenth input is the constant natural mortality flag. If true, natural mortality is a constant; otherwise it is a uniformly distributed random variable. The twentieth input is the bootstrap flag. If true, a file of bootstrapped initial population vectors is used in the projection analysis; otherwise a single initial population vector is used. Note that each of the flags must be assigned a logical value in the input file where “1" designates “true" while “0" indicates “false”.

The *biological* data are the values of a set of biological inputs needed to describe the dynamics of the age-structured population. Most of these data are required to run the AGEPRO software although some data are optional and dependent upon the simulation settings (Table 3). The biological data are read from the input file. By convention, optional inputs will be enumerated sequentially along with required inputs. Note that, if recruitment age is age-2, there is no accounting of age-1 fish in the model. However, the first element of age-specific parameter vectors is usually denoted as age-1 with subscripts (Table 3). The USER should note that this is done solely for notational convenience. In particular, if recruitment is at age-2, then the USER must input the appropriate age-2 values as the first element of the age-specific parameter vectors, such as mean spawning weights at age (see input #23 below) or fraction mature at age (see input #26 below). Again, if the recruitment age is age-2, the first element of an age-specific parameter vector corresponds to age-2 fish

and parameter values for age-1 fish should not be included in the input file.

The twenty-first input is the number of ageclasses in the population model (A), where $A \leq 25$ along with lower and upper bound on range of ages for computing mean biomass, Lowerage and Upperage. The twenty-second input is the instantaneous natural mortality rate (M), if M is constant. If M is not constant, the twenty-second input is the interval [L_M , U_M] for stochastic natural mortality. In addition, if the recruitment age is age-2 natural mortality is stochastic, then the value of M in year 0, the year immediately prior to the first year in the time horizon must also be input. The twenty-third input is the vector of mean weights at age in the stock ordered from youngest (left) to oldest (right). The twenty-fourth input is the vector of mean weights at age in the landings ordered from youngest (left) to oldest (right). If discards at age are included in the projection, the twenty-fifth input is the vector of mean weights at age of discarded fish ordered from youngest (left) to oldest (right). The twenty-sixth input is the vector of fraction mature at age ordered from youngest (left) to oldest (right). The twenty-seventh input is the fraction of total mortality that occurs prior to spawning (ZPROJ). If the total mortality flag is true, then a set of values of ZPROJ must be input. In particular, if the total mortality flag is true and the recruitment age is age-2 then the value of ZPROJ in the previous year is input first on one line followed by a line with the vector of values of ZPROJ ordered from the first (left) to the last (right) year of the time horizon is input. If the total mortality flag is false, then the constant value of ZPROJ is input, regardless of whether the recruitment age is age-2.

The twenty-eighth input is the recruitment flag which is a number from 1 to 15 that identifies the choice of stochastic stock-recruitment model to be used. These models are numbered 1 to 15 in exact correspondence with their descriptions (see **Stock-Recruitment Relationship**). The twenty-ninth input is the set of parameters needed for the chosen stock-recruitment model. The set of parameters depends on the chosen model and these are specified in Table 3 for each of the fifteen stock-recruitment models. The thirtieth input is the set of parameters to constrain recruitment for stock-recruitment models with lognormal error terms. These parameters are input only if the bounded recruitment flag is true. If this flag is true, then the endpoints of the constraining intervals are input on one line as L_{HIGH} , L_{LOW} , U_{LOW} , U_{HIGH} , while SSB_{CUT} is input on the next line. The thirty-first input is the set of parameters to define the initial population sizes for projection. The set of parameters depends on the value of the age-2 recruitment flag and the bootstrap flag, and these are specified in Table 3. The thirty-second input is the set of parameters for the prediction of a recruitment index. These parameters are input only if the index flag is true. The thirty-third input is the SSB threshold, $SSB_{THRESHOLD}$. This threshold is input only if the SSB threshold flag is true. The thirty-fourth input is the set of parameters to apply the F target option. These parameters are input only if the F target flag is true and are specified in Table 3.

The *fishery* data are the values of a set of inputs needed to describe the impact of the fishery on the population. The thirty-fifth input is the set of parameters to define fishery selectivity through time. These parameters depend upon the partial recruitment and the age-2 recruitment flag and are specified in Table 3. The thirty-sixth input is the set of parameters to define age-specific discarding through time. These parameters depend upon the discard and constant discard flags and are specified in Table 3. The thirty-seventh input is the set of parameters to define the harvest strategy. These parameters depend upon the harvest mixture, quota-based, and constant harvest strategy flags and are specified in the section Table 3. The thirty-eighth input is the set of parameters to define the market category summarization. These parameters depend upon the market category flag and are specified in Table 3.

The program SETPRO can be used to create a new input file or to create a list of the parameters in an existing input file *but this program has not yet been updated to be compatible with the current version of AGEPRO (version 2.0).* Regardless, the program may be useful for constructing a template input file for later modification. SETPRO can be run from the UNIX prompt by typing "setpro.exe" and responding to the menu choices. That is, to create an input file, one would type

```
>setpro.exe

*******************************

   1 = Create an input file

   2 = List an input file

   3 = Exit

*******************************

   Enter a choice:
1

Enter the name of the input file to be created
my_input_file

Input number: 1
Enter a title for the projection run
This_is_a_new_projection_run
```

and so on for inputs 2 through 38.

The SETPRO software checks whether the new input file exists and prompts the

USER for another filename if the input file already exists. Similarly, the software checks whether the output file already exists and prompts the USER for another filename if the output file already exists.

There are two points to note when using SETPRO to create an input file:

- Entries on a single line must be separated by a comma.
- The USER can respond to the question “Are the input data correct ?” with the ENTER key rather than typing “Y” for “yes”.

Similarly, to create a list of the parameters in an input file, one would type

```
>setpro.exe
*******************************

   1 = Create an input file

   2 = List an input file

   3 = Exit

*******************************

   Enter a choice:
2

Create a file that lists input parameters
Enter the name of the input file
my_input_filename
Enter the name of the list file
my_list_filename
```

For this example of creating a list file, a new text file named “my_list_filename” would be created. This file would contain a brief description of each of the inputs in the input file “my_input_filename”. Also note that the SETPRO software checks whether the old input file exists and prompts the USER for another filename if the input file does not exist. Similarly, the software checks whether the list file already exists and prompts the USER for another filename if the list file already exists.

Model Outputs

The AGEPRO software creates an output file that summarizes the projection analysis results. The first five are descriptive and are always in the output file. The first output is the character string describing the run (*Simulation Input #1*). The second and third outputs are the names of the input and output files for the run (*System Data*). The fourth output is the number of the recruitment model (*Biological Input #28*). The fifth output is the number of simulations per initial population vector (*Simulation Input #4*).

The sixth output describes the harvest strategy applied to the stock during each year in the time horizon. If the harvest strategy is quota-based, then the annual quotas are listed. If the harvest strategy is F-based, then the annual Fs are listed. If the harvest strategy is a mixture of quotas and Fs, then the quota or F is listed for each year. The sixth output is always part of the output file.

The seventh output describes the SSB trajectory through time. Average SSB and the standard deviation of SSB by year are reported. Percentiles of the empirical distribution of realized SSB levels by year are also reported. The seventh output is always in the output file. The eighth output gives the probability that the SSB threshold was exceeded in each year. The eighth output is in the output file only if the SFA threshold flag is true.

The ninth output describes the mean biomass (MB) trajectory through time. Average MB values and the standard deviation of MB by year are reported. Percentiles of the empirical distribution of realized MB levels by year are also reported. The tenth output gives the probability that the MB threshold was exceeded in each year. The tenth output is in the output file only if the SFA threshold flag is true.

The eleventh output describes the fishing mortality weighted by biomass (Fwb) trajectory through time. Average Fwb values and the standard deviation of Fwb by year are reported. Percentiles of the empirical distribution of realized Fwb levels by year are also reported. The twelfth output gives the probability that the Fwb threshold was exceeded in each year. The twelfth output is in the output file only if the SFA threshold flag is true.

The thirteenth output describes the total stock biomass (TSB) trajectory through time. Average TSB values and the standard deviation of TSB by year are reported. Percentiles of the empirical distribution of realized TSB levels by year are also reported. The fourteenth output gives the probability that the TSB threshold was exceeded in each year. The fourteenth output is in the output file only if the SFA threshold flag is true.

The fifteenth output describes the recruitment trajectory through time. Average recruitment and the standard deviation of recruitment by birth year of the yearclass are reported. Here it is important to note that the recruitment output for year t is the number of age-R fish in year t+R where R is the age of recruitment (age-1 or age-2). Percentiles of

the empirical distribution of realized recruitment levels by birth year of the yearclass are also reported. The fifteenth output is always in the output file.

The sixteenth output describes the predictions of a catch-at-age index through time. Averages of the estimated index and the standard deviation of the estimated index by year are reported. Percentiles of the empirical distribution of the estimated index by year are also reported. The sixteenth output also gives the probability that the predicted index exceeded a USER-specified critical value in each year. The sixteenth output is in the output file only if the index flag is true.

The seventeenth output describes realized landings through time under an F-based harvest strategy. Average landings and the standard deviation of landings by year are reported. Percentiles of the empirical distribution of landings by year are also reported. The seventeenth output is in the output file if either the quota-based flag is true or the harvest mixture flag is true.

The eighteenth output describes landings by market category in each year. Average landings in weight and number of fish and the standard deviation of these landings by year are reported for each market category. Percentiles of the empirical distribution of landings in weight and number of fish by year are also reported for each market category. The eighteenth output is in the output file only if the market category flag is true. Note also that if the market category flag is true, a separate file containing the individual realizations of the landings by market category is output. The structure of the market category file is as follows. The first part of the market category file lists the average total weight (kg) and numbers of fish by year followed by the median total weight (kg) and numbers of fish by year for each market category. The list for market category 1 appears first, followed by the list for market category 2, and the list for market category 3, if necessary. The second part of the market category file gives the realized set of landings in weight by year for each simulation with the landings broken out by market category. For each year in the time horizon, there is one line in the market category file for each simulated landing. The first entry on the line is the total landings by weight followed by the amount landed in market category 1, market category 2, and market category 3. All landed weights are reported in kilograms. Note that the market category file can require substantial disk storage space if the number of initial population sizes and simulations is large.

The nineteenth output describes the total weight discarded through time. Averages of discards and the standard deviation of discards by year are reported. Percentiles of the empirical distribution of the discards by year are also reported. The nineteenth output is in the output file only if the discard flag is true.

The twentieth output describes realized F through time under a quota-based harvest strategy. Average F and the standard deviation of F by year are reported. Percentiles of the empirical distribution of F by year are also reported. The twentieth output is in the output file if either the quota-based flag is true, the F target flag is true, or the harvest mixture flag is true. The twenty-first output gives the probability that the fully-recruited fishing

mortality threshold was exceeded in each year. The twenty-first output is in the output file only if the SFA threshold flag is true.

Program Structure

The AGEPRO software is a structured program originally written in Fortran 77 (Metcalf 1985) and later translated to Fortran 90 (Metcalf and Reid 1998). The source code is provided with this USER'S GUIDE (Appendix 2). The program uses standard Fortran 77 syntax along with Fortran 90 syntax with one exception. The exception is the use of the *do while-end do* syntax for looping. Although the *do while-end do* syntax is not standard Fortran 77, most compilers will accept this syntax. For example, the Fortran compilers (f77) found on the SUN OS 4.3 and the SGI IRIX 5.3 operating systems accept the *do while-end do* syntax.

The AGEPRO software is structured into 8 sections: *variable declarations, read system data, read input data, initialize variables, projection, summarize results, output results, function and subroutine declarations*. These sections are identified in the source code (Appendix 2). Brief descriptions of two sections, *projection* and *function and subroutine declarations* are provided below.

The *projection* section consists of 3 nested loops. The outer loop is over the number of initial population vectors (IC loop). The middle loop is over the number of simulations to perform for each initial population vector (SIM loop). The inner loop is over the number of time periods within the time horizon (TIME loop). The *projection* section ends when all loops have been completed. There can be up to 500 initial population vectors, 200 simulations per initial population vector, and 25 time periods for a maximum of 2.5 million loops. Note that if a harvest quota is too large to be taken from the available population size at any time within the time loop, the time loop is exited and that particular combination of initial population vector and simulation is marked as being infeasible for the summarization of results.

Excluding the memory allocation subroutines, the *function and subroutine declarations* section consists of 3 functions and 12 subroutines. The 3 functions are taken from Press *et al.* (1992); these are *gasdev()*, *ran2()*, and *rtsafe()*. The function *gasdev()* generates a standard unit normal random deviate. The function *ran2()* generates a uniformly distributed random variable on the interval [0,1]. The function *rtsafe()* numerically computes the root of a given equation specified with the subroutine *funcd()*. The 12 subroutines are *calc_catch()*, *calc_next()*, *calc_ssb()*, *calc_meanB(), calc_totB(), calc_FmeanB(), funcd()*, *hpsort()*, *simavg()*, *simsd()*, *summarize()*, and *warmup()*. The subroutine *cal_catch()* computes the catch at age for a given F and population size. The subroutine *calc_next()* computes the population vector in the next time period given the current population vector, recruitment, and F. The subroutine *calc_ssb()* computes spawning stock biomass for a given population vector and F. The subroutine *calc_meanB()*

computes mean stock biomass for a given population vector and F. The subroutine *calc_totB()* computes total stock biomass for a given population vector and F. The subroutine *calc_FmeanB()* computes fishing mortality weighted by biomass for a given population vector and F. The subroutine *funcd()* computes the value and derivative of the function L(F)-Q for a given value of F and quota Q. The subroutine *hpsort()* sorts an array and is taken from Press et al. (1992). The subroutine *simavg()* computes the average of the feasible values for model outputs such as SSB and recruitment. The subroutine *simsd()* computes the standard deviation of the feasible values for model outputs such as SSB and recruitment. The subroutine *summarize()* computes averages, standard deviations, and percentiles of model outputs. The subroutine *warmup()* initializes the starting point in the sequence of pseudo-random numbers generated with the function *ran2()*.

Examples

Three examples of projection runs are presented below to illustrate general features of the AGEPRO model. Note that these examples are hypothetical. Further, they were created using the version 1.21 AGEPRO software.

Example 1 (example based on AGEPRO version 1.21)

The first example is taken from a hypothetical projection analysis for yellowtail flounder. This example illustrates a projection with age-2 recruitment, an initial distribution of population vectors, and discards at age. The projection begins in 1994 with a time horizon of 12 years. There are 5 simulations performed for each of 500 initial population vectors. The recruitment age is age-2 and harvest is F-based and time-varying. Realized Ssb levels are compared to a threshold and natural mortality is constant. The stock-recruitment model is Beverton-Holt with lognormal error. Below are the list file created with the SETPRO software and the output file created with the AGEPRO software.

List of Input File for Example 1

```
TITLE OF THE PROJECTION RUN
flounder_example1

FIRST YEAR OF THE PROJECTION RUN
     1994

LENGTH OF THE TIME HORIZON
       12

NUMBER OF SIMULATIONS PER INITIAL POPULATION SIZE
        5

POSITIVE SEED FOR RANDOM NUMBER GENERATION
```

AGE OF RECRUITMENT
RECRUITMENT AT AGE-2

MIXED HARVEST STRATEGY
NO MIXTURE OF FISHING MORTALITIES AND QUOTAS

DISCARDS
DISCARDS AT AGE INCLUDED

TYPE OF HARVEST
HARVEST SET BY FISHING MORTALITY RATE

HARVEST STRATEGY
TIME-VARYING

TARGET F
TARGET F IS NOT INCLUDED

SURVEY CATCH-AT-AGE INDEX
NO SURVEY INDEX IS PREDICTED

SSB THRESHOLD
SSB IS COMPARED TO THRESHOLD

LANDINGS BY MARKET CATEGORY
LANDINGS ARE NOT SUMMARIZED BY CATEGORY

FRACTION OF TOTAL MORTALITY PRIOR TO SPAWNING
CONSTANT IN TIME

PARTIAL RECRUITMENT
CONSTANT IN TIME

DISCARD FRACTION AT AGE
CONSTANT IN TIME

BOUNDED RECRUITMENT
NO R/SSB CONSTRAINTS ARE USED

NATURAL MORTALITY
NATURAL MORTALITY IS CONSTANT

INITIAL POPULATION SIZE
DISTRIBUTION OF INITIAL POPULATION SIZES

NUMBER OF AGE CLASSES
5

CONSTANT NATURAL MORTALITY
0.200000

MEAN WEIGHTS AT AGE
0.315000 0.393000 0.526000 0.656000 0.939000

MEAN LANDED WEIGHTS AT AGE
0.315000 0.393000 0.526000 0.656000 0.939000

MEAN DISCARDED WEIGHTS AT AGE
0.157000 0.256000 0.313000 0.656000 0.939000

FRACTION MATURE AT AGE
0.880000 1.000000 1.000000 1.000000 1.000000

FRACTION OF TOTAL MORTALITY BEFORE SPAWNING
0.416670

STOCHASTIC RECRUITMENT MODEL
5

BEVERTON-HOLT WITH LOGNORMAL ERROR MODEL

PARAMETERS A, B, AND RESIDUAL VARIANCE
32943.605469 10122.372070 0.470113

CONVERSION COEFFICIENTS FOR SSB AND RECRUITMENT
1000.000 1000.000

DISTRIBUTION OF INITIAL POPULATION SIZE

NUMBER OF INITIAL POPULATION VECTORS
500

FILE WITH INITIAL POPULATION VECTORS
gbyt94n2.dat

CONVERSION COEFFICIENT

1000000.000

FILE WITH FISHING MORTALITY IN PREVIOUS YEAR
gbyt94ff.dat

SSB THRESHOLD
10000000.000

CONSTANT PARTIAL RECRUITMENT
PARTIAL RECRUITMENT AT AGE
0.140000 0.510000 1.000000 1.000000 1.000000

CONSTANT DISCARD FRACTION AT AGE
0.583000 0.261000 0.072000 0.000000 0.000000

HARVEST SET BY FISHING MORTALITY
TIME-VARYING FISHING MORTALITY BY YEAR
1.080000 0.960000 0.840000 0.720000 0.600000 0.600000 0.600000
0.600000 0.600000 0.600000 0.600000 0.600000

Output File for Example 1

PROJECTION RUN: flounder_example1
INPUT FILE: example1
OUTPUT FILE: example1.out
RECRUITMENT MODEL: 5
NUMBER OF SIMULATIONS: 5

F-BASED PROJECTIONS
TIME-VARYING F
YEAR F
1994 1.080
1995 0.960
1996 0.840
1997 0.720
1998 0.600
1999 0.600
2000 0.600
2001 0.600
2002 0.600
2003 0.600
2004 0.600
2005 0.600

SPAWNING STOCK BIOMASS (THOUSAND MT)

YEAR	AVG SSB (000 MT)	STD
1994	5.101	2.376
1995	6.128	2.779
1996	7.416	3.751
1997	8.899	4.595
1998	10.980	5.756
1999	13.049	6.538
2000	14.957	7.332
2001	16.668	7.937
2002	18.306	8.621
2003	19.481	8.982
2004	20.507	9.063
2005	21.403	9.403

PERCENTILES OF SPAWNING STOCK BIOMASS (000 MT)

YEAR	1%	5%	10%	25%	50%	75%	90%	95%	99%
1994	2.067	2.428	2.725	3.439	4.492	6.099	8.232	9.690	13.420
1995	2.260	2.876	3.233	4.165	5.526	7.459	9.690	11.370	15.750
1996	2.490	3.244	3.696	4.895	6.600	8.948	12.003	14.371	20.336
1997	2.757	3.641	4.348	5.700	7.837	10.849	14.548	17.315	25.091
1998	3.312	4.428	5.243	7.030	9.841	13.414	17.906	21.854	30.584
1999	3.881	5.316	6.370	8.542	11.700	15.938	21.654	25.202	34.148
2000	4.385	6.236	7.463	9.825	13.437	18.157	24.382	28.676	39.935
2001	4.925	7.330	8.540	11.083	15.031	20.322	26.839	31.169	43.905
2002	5.489	7.987	9.302	12.349	16.577	22.513	28.951	33.436	47.962
2003	6.417	8.594	10.009	13.277	17.731	23.590	31.040	35.932	48.168
2004	6.869	9.325	10.933	14.140	18.814	25.143	32.520	37.044	49.945
2005	7.197	10.033	11.566	14.755	19.522	25.978	33.108	38.824	50.394

ANNUAL PROBABILITY THAT SSB EXCEEDS THRESHOLD: 10.00000 THOUSAND MT

YEAR	Pr(SSB > Threshold Value)
1994	0.044
1995	0.085

1996	0.180
1997	0.315
1998	0.490
1999	0.626
2000	0.736
2001	0.819
2002	0.871
2003	0.901
2004	0.933
2005	0.950

RECRUITMENT UNITS ARE: 1000000. FISH

BIRTH YEAR	AVG RECRUITMENT	STD
1994	10.049	8.157
1995	13.242	11.344
1996	14.704	12.310
1997	17.057	15.298
1998	18.805	15.258
1999	20.668	17.354
2000	22.322	18.708
2001	24.177	20.117
2002	24.288	20.911
2003	25.506	19.824
2004	26.336	21.388
2005	26.760	20.049

PERCENTILES OF RECRUITMENT UNITS ARE: 1000000. FISH

BIRTH YEAR	1%	5%	10%	25%	50%	75%	90%	95%	99%
1994	1.444	2.422	3.198	4.854	7.724	12.589	19.332	25.640	41.651
1995	1.869	2.944	3.917	6.177	10.358	16.397	25.731	33.256	57.732
1996	2.101	3.441	4.380	6.940	11.332	18.539	28.340	36.187	67.485
1997	2.345	3.804	5.093	7.763	13.065	21.112	32.158	42.134	75.992
1998	2.364	4.082	5.397	8.810	14.356	23.990	37.089	49.613	76.260
1999	3.015	4.872	6.222	9.707	15.861	26.014	40.548	50.414	86.269
2000	3.094	5.263	6.966	10.743	17.225	27.580	42.758	54.902	91.411

2001	3.448	5.604	7.356	11.581	18.708	30.349	47.027	59.455	102.192
2002	3.248	5.859	7.399	11.471	18.590	30.351	45.832	62.045	103.841
2003	3.716	6.169	8.017	12.037	19.977	32.229	49.969	63.617	96.740
2004	4.014	6.778	8.400	12.894	20.412	32.844	51.225	66.255	100.983
2005	3.881	6.549	8.462	13.223	21.467	33.395	51.896	66.971	102.840

LANDINGS FOR F-BASED PROJECTIONS

YEAR	AVG LANDINGS (000 MT)	STD
1994	1.457	0.390
1995	1.848	0.740
1996	2.372	1.064
1997	2.481	1.159
1998	2.748	1.371
1999	3.470	1.729
2000	4.155	2.066
2001	4.750	2.257
2002	5.294	2.475
2003	5.787	2.660
2004	6.194	2.810
2005	6.495	2.874

PERCENTILES OF LANDINGS (000 MT)

YEAR	1%	5%	10%	25%	50%	75%	90%	95%	99%
1994	0.762	0.920	0.991	1.170	1.412	1.710	1.978	2.168	2.539
1995	0.825	0.974	1.092	1.336	1.672	2.192	2.816	3.281	4.374
1996	0.860	1.111	1.262	1.603	2.138	2.922	3.787	4.401	5.758
1997	0.866	1.115	1.282	1.675	2.233	3.004	3.924	4.663	6.341
1998	0.865	1.163	1.359	1.806	2.461	3.369	4.394	5.283	7.211
1999	1.060	1.458	1.726	2.269	3.130	4.263	5.538	6.681	9.481
2000	1.272	1.730	2.057	2.729	3.758	5.053	6.716	7.921	11.070
2001	1.458	2.033	2.399	3.159	4.290	5.765	7.800	8.998	12.613
2002	1.656	2.336	2.721	3.604	4.808	6.447	8.486	9.689	14.020
2003	1.765	2.562	3.052	3.948	5.234	7.117	9.049	10.550	15.027
2004	2.037	2.805	3.235	4.236	5.692	7.496	9.718	11.094	15.337
2005	2.178	2.991	3.472	4.505	5.983	7.859	10.206	11.698	15.389

DISCARDS FOR F-BASED PROJECTIONS

YEAR	AVG DISCARDS (000 MT)	STD
1994	0.262	0.118

1995	0.358	0.179
1996	0.334	0.174
1997	0.338	0.196
1998	0.340	0.196
1999	0.393	0.228
2000	0.438	0.241
2001	0.478	0.264
2002	0.519	0.286
2003	0.549	0.302
2004	0.566	0.308
2005	0.587	0.307

PERCENTILES OF DISCARDS (000 MT)

YEAR	1%	5%	10%	25%	50%	75%	90%	95%	99%
1994	0.095	0.128	0.142	0.179	0.233	0.307	0.423	0.492	0.679
1995	0.111	0.148	0.174	0.229	0.314	0.448	0.601	0.700	0.917
1996	0.104	0.138	0.162	0.215	0.295	0.408	0.546	0.664	0.945
1997	0.089	0.127	0.150	0.208	0.294	0.414	0.575	0.710	1.008
1998	0.087	0.125	0.152	0.208	0.296	0.417	0.571	0.701	1.047
1999	0.104	0.145	0.177	0.240	0.342	0.485	0.663	0.813	1.165
2000	0.118	0.165	0.197	0.273	0.383	0.537	0.735	0.909	1.303
2001	0.130	0.184	0.223	0.302	0.417	0.590	0.801	0.952	1.438
2002	0.141	0.201	0.240	0.326	0.453	0.640	0.861	1.021	1.524
2003	0.157	0.213	0.255	0.343	0.487	0.670	0.916	1.075	1.590
2004	0.166	0.226	0.268	0.361	0.500	0.680	0.941	1.132	1.591
2005	0.174	0.237	0.281	0.374	0.522	0.726	0.963	1.142	1.628

Example 2 (example based on AGEPRO version 1.21)

The second example is also taken from a hypothetical projection analysis for yellowtail flounder. This example illustrates a projection with age-2 recruitment, a single initial population vector, stochastic natural mortality, time-varying total mortality prior to spawning, time-varying partial recruitment, and market category summaries. As in example 1, the projection begins in 1994 with a time horizon of 12 years. There are 200 simulations performed for a single initial population vector. In contrast to example 1, natural mortality is random, the partial recruitment vector changes in the second year, and the fraction of total mortality prior to spawning changes from 0.3 in 1993 to 0.4167 in 1994. Summaries of landings in 2 market categories are listed but the raw data are not for brevity. Below are the list file and output file.

List of Input File for Example 2

TITLE OF THE PROJECTION RUN
flounder_example2

FIRST YEAR OF THE PROJECTION RUN
1994

LENGTH OF THE TIME HORIZON
12

NUMBER OF SIMULATIONS PER INITIAL POPULATION SIZE
200

POSITIVE SEED FOR RANDOM NUMBER GENERATION
5190

AGE OF RECRUITMENT
RECRUITMENT AT AGE-2

MIXED HARVEST STRATEGY
NO MIXTURE OF FISHING MORTALITIES AND QUOTAS

DISCARDS
DISCARDS AT AGE INCLUDED

TYPE OF HARVEST
HARVEST SET BY FISHING MORTALITY RATE

HARVEST STRATEGY
TIME-VARYING

TARGET F
TARGET F IS NOT INCLUDED

SURVEY CATCH-AT-AGE INDEX
NO SURVEY INDEX IS PREDICTED

SSB THRESHOLD
SSB IS COMPARED TO THRESHOLD

LANDINGS BY MARKET CATEGORY
LANDINGS ARE SUMMARIZED BY CATEGORY

FRACTION OF TOTAL MORTALITY PRIOR TO SPAWNING
TIME-VARYING

PARTIAL RECRUITMENT
TIME-VARYING

DISCARD FRACTION AT AGE
CONSTANT IN TIME

BOUNDED RECRUITMENT
NO R/SSB CONSTRAINTS ARE USED

NATURAL MORTALITY
NATURAL MORTALITY IS A RANDOM VARIABLE

INITIAL POPULATION SIZE
SINGLE INITIAL POPULATION SIZE

NUMBER OF AGE CLASSES
5

LOWER AND UPPER BOUND FOR RANDOM NATURAL MORTALITY
0.180000 0.220000

NATURAL MORTALITY IN PREVIOUS YEAR
0.200000

MEAN WEIGHTS AT AGE
0.315000 0.393000 0.526000 0.656000 0.939000

MEAN LANDED WEIGHTS AT AGE
0.315000 0.393000 0.526000 0.656000 0.939000

MEAN DISCARDED WEIGHTS AT AGE
0.157000 0.256000 0.313000 0.656000 0.939000

FRACTION MATURE AT AGE
0.880000 1.000000 1.000000 1.000000 1.000000

FRACTION OF TOTAL MORTALITY BEFORE SPAWNING
IN PREVIOUS YEAR
0.300000

FRACTION OF TOTAL MORTALITY BEFORE SPAWNING

BY YEAR
0.416670 0.416670 0.416670 0.416670 0.416670 0.416670 0.416670 0.416670 0.416670 0.416670 0.416670 0.416670 0.416670

STOCHASTIC RECRUITMENT MODEL
5

BEVERTON-HOLT WITH LOGNORMAL ERROR MODEL

PARAMETERS A, B, AND RESIDUAL VARIANCE
32943.605469 10122.372070 0.470113

CONVERSION COEFFICIENTS FOR SSB AND RECRUITMENT
1000.000 1000.000

SINGLE INITIAL POPULATION SIZE

CONVERSION COEFFICIENT
1000000.000

INITIAL POPULATION VECTOR
14.463 1.676 1.843 0.563 0.136

FISHING MORTALITY IN PREVIOUS YEAR
1.223018

SSB THRESHOLD
10000000.000

TIME-VARYING PARTIAL RECRUITMENT
PARTIAL RECRUITMENT AT AGE IN PREVIOUS YEAR
0.100000 0.500000 1.000000 1.000000 1.000000

PARTIAL RECRUITMENT AT AGE IN YEAR: 1
0.050000 0.350000 1.000000 1.000000 1.000000

PARTIAL RECRUITMENT AT AGE IN YEAR: 2
0.050000 0.350000 1.000000 1.000000 1.000000

PARTIAL RECRUITMENT AT AGE IN YEAR: 3
0.050000 0.350000 1.000000 1.000000 1.000000

PARTIAL RECRUITMENT AT AGE IN YEAR: 4
0.050000 0.350000 1.000000 1.000000 1.000000

PARTIAL RECRUITMENT AT AGE IN YEAR: 5
  0.050000 0.350000 1.000000 1.000000 1.000000

PARTIAL RECRUITMENT AT AGE IN YEAR: 6
  0.050000 0.350000 1.000000 1.000000 1.000000

PARTIAL RECRUITMENT AT AGE IN YEAR: 7
  0.050000 0.350000 1.000000 1.000000 1.000000

PARTIAL RECRUITMENT AT AGE IN YEAR: 8
  0.050000 0.350000 1.000000 1.000000 1.000000

PARTIAL RECRUITMENT AT AGE IN YEAR: 9
  0.050000 0.350000 1.000000 1.000000 1.000000

PARTIAL RECRUITMENT AT AGE IN YEAR: 10
  0.050000 0.350000 1.000000 1.000000 1.000000

PARTIAL RECRUITMENT AT AGE IN YEAR: 11
  0.050000 0.350000 1.000000 1.000000 1.000000

PARTIAL RECRUITMENT AT AGE IN YEAR: 12
  0.050000 0.350000 1.000000 1.000000 1.000000

CONSTANT DISCARD FRACTION AT AGE
  0.583000 0.261000 0.072000 0.000000 0.000000

HARVEST SET BY FISHING MORTALITY
TIME-VARYING FISHING MORTALITY BY YEAR
  1.080000 0.960000 0.840000 0.720000 0.600000 0.600000 0.600000
0.600000 0.600000 0.600000 0.600000 0.600000

SUMMARIZE LANDINGS BY MARKET CATEGORY
NUMBER OF MARKET CATEGORIES
     2

PROPORTION LANDED AT AGE IN CATEGORY: 1
  0.817000 0.692000 0.363000 0.203000 0.000000

PROPORTION LANDED AT AGE IN CATEGORY: 2
  0.183000 0.308000 0.637000 0.797000 1.000000

FILE WITH MARKET CATEGORY DATA

gbyt1.mkt

Output File for Example 2

PROJECTION RUN: flounder_example2
INPUT FILE: example2
OUTPUT FILE: example2.out
RECRUITMENT MODEL: 5
NUMBER OF SIMULATIONS: 200

F-BASED PROJECTIONS
TIME-VARYING F

YEAR	F
1994	1.080
1995	0.960
1996	0.840
1997	0.720
1998	0.600
1999	0.600
2000	0.600
2001	0.600
2002	0.600
2003	0.600
2004	0.600
2005	0.600

SPAWNING STOCK BIOMASS (THOUSAND MT)

YEAR	AVG SSB (000 MT)	STD
1994	4.984	0.024
1995	6.624	2.031
1996	8.489	3.731
1997	10.290	4.866
1998	12.760	5.744
1999	15.268	6.826
2000	17.478	7.596
2001	19.904	8.394
2002	21.667	8.968
2003	23.329	10.424
2004	24.751	10.883
2005	25.816	10.919

PERCENTILES OF SPAWNING STOCK BIOMASS (000 MT)

YEAR	1%	5%	10%	25%	50%	75%	90%	95%	99%

1994 4.943 4.948 4.950 4.964 4.982 5.005 5.018 5.022 5.025
1995 4.488 4.752 4.939 5.356 6.021 7.339 8.511 9.736 14.892
1996 4.127 4.481 5.092 6.037 7.822 9.606 12.465 15.353 20.424
1997 3.307 4.620 5.334 7.051 9.100 12.674 16.603 18.391
24.120
1998 4.184 5.870 6.700 8.762 11.841 15.512 19.420 23.498
34.398
1999 4.963 6.794 8.451 11.017 13.609 17.643 24.786 30.410
35.495
2000 5.781 7.817 9.384 12.108 15.987 21.182 27.735 32.892
39.991
2001 7.996 9.361 10.646 13.507 18.215 24.222 31.543 34.770
47.090
2002 7.520 10.331 11.245 15.184 20.633 27.157 33.389 37.740
48.420
2003 8.850 10.971 12.439 15.974 21.384 28.237 35.604 40.755
58.248
2004 10.016 12.140 13.503 16.900 22.529 29.501 38.220 43.174
58.976
2005 10.320 12.082 14.304 17.662 23.697 31.179 39.980 44.991
62.775

ANNUAL PROBABILITY THAT SSB EXCEEDS THRESHOLD: 10.00000
THOUSAND MT
YEAR Pr(SSB > Threshold Value)
1994 0.000
1995 0.050
1996 0.215
1997 0.420
1998 0.635
1999 0.795
2000 0.885
2001 0.935
2002 0.955
2003 0.970
2004 0.995
2005 0.995

RECRUITMENT UNITS ARE: 1000000. FISH
BIRTH
YEAR AVG RECRUITMENT STD
1994 9.883 8.113
1995 14.044 11.284
1996 15.361 12.254

1997	18.039	13.663
1998	20.356	16.466
1999	21.711	16.485
2000	25.416	19.779
2001	25.080	17.496
2002	27.320	27.206
2003	28.269	22.100
2004	28.338	18.561
2005	28.572	21.210

PERCENTILES OF RECRUITMENT UNITS ARE: 1000000. FISH

BIRTH YEAR	1%	5%	10%	25%	50%	75%	90%	95%	99%
1994	1.464	2.535	3.137	4.767	7.550	12.821	17.230	22.654	43.317
1995	1.934	2.777	4.298	7.255	10.757	18.104	25.781	34.297	64.112
1996	2.150	3.657	4.714	7.368	11.171	20.150	30.844	38.752	53.995
1997	2.851	4.978	6.000	9.635	14.184	22.804	34.305	39.712	78.797
1998	3.094	4.358	5.867	10.131	15.622	25.126	37.851	46.504	84.413
1999	2.503	5.418	7.317	10.450	15.730	28.904	43.048	52.423	84.092
2000	3.386	6.568	8.467	11.721	20.153	31.870	47.577	62.248	109.176
2001	3.875	6.954	8.629	12.267	19.316	34.384	48.241	62.435	76.902
2002	3.334	5.301	7.179	11.860	20.626	32.172	56.594	69.405	122.876
2003	5.599	7.312	8.952	12.782	22.136	35.885	49.996	83.145	99.207
2004	3.842	6.715	9.085	14.663	23.165	38.168	55.305	63.093	89.206
2005	4.358	7.381	9.746	14.611	23.285	35.053	53.380	67.096	115.016

LANDINGS FOR F-BASED PROJECTIONS

YEAR	AVG LANDINGS (000 MT)	STD
1994	1.081	0.005
1995	1.459	0.050
1996	2.474	0.441
1997	2.594	1.110

1998	2.980	1.415
1999	3.771	1.753
2000	4.566	2.039
2001	5.268	2.364
2002	5.911	2.533
2003	6.564	2.844
2004	6.995	2.930
2005	7.472	3.593

PERCENTILES OF LANDINGS (000 MT)

YEAR	1%	5%	10%	25%	50%	75%	90%	95%	99%
1994	1.072	1.073	1.074	1.077	1.081	1.086	1.088	1.089	1.090
1995	1.395	1.401	1.407	1.426	1.451	1.477	1.509	1.534	1.627
1996	1.994	2.051	2.109	2.193	2.353	2.647	2.866	3.166	4.193
1997	1.277	1.477	1.612	1.883	2.321	3.001	3.695	4.562	6.157
1998	1.049	1.362	1.614	2.086	2.687	3.447	4.588	5.690	7.284
1999	1.153	1.684	1.939	2.571	3.359	4.681	5.790	6.768	8.775
2000	1.493	2.023	2.464	3.276	4.084	5.399	7.127	8.695	10.876
2001	1.869	2.366	2.882	3.702	4.714	5.978	8.204	9.930	12.961
2002	2.297	2.864	3.222	3.960	5.171	7.284	9.331	10.676	12.719
2003	2.220	2.963	3.350	4.530	5.978	8.015	10.014	11.339	16.706
2004	2.489	3.354	3.658	4.832	6.498	8.522	10.764	12.290	16.154
2005	2.987	3.472	3.915	4.971	6.673	8.838	11.546	13.355	17.682

LANDINGS BY MARKET CATEGORY

MARKET CATEGORY: 1

YEAR	AVG LANDINGS (000 MT)	STD
1994	0.415	0.002
1995	0.765	0.039
1996	1.045	0.306
1997	1.040	0.528
1998	1.088	0.575
1999	1.318	0.651
2000	1.520	0.745
2001	1.695	0.843
2002	1.877	0.923
2003	2.031	0.982
2004	2.131	1.056
2005	2.242	1.249

MARKET CATEGORY: 1

YEAR	AVG LANDINGS (000 FISH)	STD
1994	928.810	4.721

1995	1875.120	121.146
1996	2281.546	787.275
1997	2278.754	1206.816
1998	2366.725	1241.198
1999	2834.620	1396.589
2000	3254.889	1619.958
2001	3611.850	1786.244
2002	4010.157	2000.903
2003	4297.657	2039.298
2004	4513.362	2360.428
2005	4743.807	2590.643

PERCENTILES OF LANDINGS (000 MT)
MARKET CATEGORY: 1

YEAR	1%	5%	10%	25%	50%	75%	90%	95%	99%
1994	0.412	0.412	0.412	0.414	0.415	0.417	0.418	0.418	0.419
1995	0.721	0.725	0.728	0.740	0.756	0.781	0.805	0.830	0.908
1996	0.711	0.757	0.787	0.852	0.959	1.163	1.313	1.533	2.241
1997	0.390	0.491	0.563	0.700	0.934	1.220	1.561	2.037	2.768
1998	0.264	0.413	0.495	0.709	0.942	1.340	1.797	2.135	2.973
1999	0.365	0.540	0.636	0.870	1.157	1.637	2.007	2.524	3.627
2000	0.452	0.612	0.835	1.013	1.364	1.762	2.390	3.017	4.127
2001	0.570	0.716	0.789	1.116	1.507	2.099	2.731	3.228	5.012
2002	0.496	0.762	0.936	1.251	1.632	2.241	3.082	3.527	4.502
2003	0.542	0.771	0.958	1.324	1.843	2.477	3.224	3.908	5.462
2004	0.659	0.916	1.112	1.354	1.898	2.614	3.474	3.886	5.404
2005	0.775	0.947	1.060	1.407	1.884	2.756	3.679	4.172	5.972

PERCENTILES OF LANDINGS (000 FISH)
MARKET CATEGORY: 1

YEAR	1%	5%	10%	25%	50%	75%	90%	95%	99%
1994	920.919	921.805	922.185	924.812	928.430	932.810	935.379	936.042	936.663
1995	1741.642	1755.352	1766.523	1801.597	1845.567	1915.183	1993.613	2082.715	2334.687
1996	1406.098	1541.898	1607.554	1782.796	2072.607	2573.782	2993.823	3647.390	5323.954
1997	774.332	1000.465	1177.000	1485.790	2034.064	2721.938	3454.620	4567.753	5959.673
1998	565.903	933.153	1127.740	1517.593	2050.550	2968.315	3841.042	4484.551	5887.051
1999	782.870	1156.023	1379.373	1862.583	2508.675	3492.652	4370.167	5312.497	8121.758
2000	993.709	1288.203	1781.391	2220.790	2862.990	3792.414	5352.012		

6787.442 8464.320
2001 1219.902 1530.043 1796.679 2375.424 3187.060 4448.634 5893.870 6827.436 10057.199
2002 1018.379 1552.973 1908.710 2664.994 3601.091 4950.250 6327.266 7889.650 10688.919
2003 1210.099 1798.381 2062.266 2752.791 3901.262 5379.981 6876.431 8158.952 10815.159
2004 1417.925 1868.046 2308.837 2917.120 3893.534 5473.977 7398.915 7919.268 10834.707
2005 1634.380 1972.436 2233.185 3061.979 3952.981 5584.436 8112.084 8782.771 13804.104

MARKET CATEGORY: 2

YEAR	AVG LANDINGS (000 MT)	STD
1994	0.666	0.003
1995	0.693	0.012
1996	1.429	0.136
1997	1.554	0.614
1998	1.892	0.873
1999	2.453	1.156
2000	3.046	1.369
2001	3.573	1.604
2002	4.033	1.731
2003	4.533	1.947
2004	4.865	2.018
2005	5.230	2.453

MARKET CATEGORY: 2

YEAR	AVG LANDINGS (000 FISH)	STD
1994	1182.031	5.634
1995	1354.995	32.220
1996	2714.869	344.661
1997	2814.398	1215.545
1998	3255.064	1625.964
1999	4144.996	2012.682
2000	5024.945	2319.645
2001	5815.103	2725.366
2002	6501.709	2893.114
2003	7267.254	3282.735
2004	7729.674	3293.217
2005	8259.064	4174.600

PERCENTILES OF LANDINGS (000 MT)
MARKET CATEGORY: 2

YEAR	1%	5%	10%	25%	50%	75%	90%	95%	99%
1994	0.661	0.661	0.661	0.663	0.666	0.669	0.670	0.671	0.671
1995	0.673	0.676	0.679	0.685	0.692	0.701	0.708	0.714	0.729
1996	1.281	1.300	1.311	1.342	1.393	1.474	1.567	1.660	1.952
1997	0.845	0.969	1.019	1.170	1.402	1.781	2.108	2.704	3.871
1998	0.786	0.944	1.088	1.324	1.701	2.225	2.795	3.448	4.727
1999	0.748	1.086	1.308	1.654	2.167	2.979	3.766	4.449	5.558
2000	0.986	1.424	1.584	2.143	2.799	3.576	4.721	5.662	8.255
2001	1.153	1.500	1.988	2.557	3.166	4.062	5.699	7.035	8.466
2002	1.457	1.949	2.169	2.736	3.590	4.849	6.370	7.270	8.801
2003	1.678	2.002	2.313	3.181	4.205	5.491	6.887	7.951	11.361
2004	1.812	2.277	2.608	3.361	4.600	6.003	7.291	8.611	10.657
2005	2.106	2.378	2.734	3.513	4.814	6.299	8.008	9.054	12.463

PERCENTILES OF LANDINGS (000 FISH)
MARKET CATEGORY: 2

YEAR	1%	5%	10%	25%	50%	75%	90%	95%	99%
1994	1172.612	1173.671	1174.124	1177.260	1181.579	1186.806	1189.871	1190.662	1191.402
1995	1307.858	1311.904	1318.807	1332.866	1351.122	1368.344	1391.374	1404.760	1452.184
1996	2344.650	2385.210	2423.289	2496.917	2618.715	2840.517	3037.254	3276.493	4054.676
1997	1393.093	1627.636	1748.561	2046.781	2474.882	3270.554	3961.915	4894.660	7193.100
1998	1150.842	1477.036	1751.530	2213.630	2842.441	3872.901	4988.354	5995.794	8649.986
1999	1195.424	1766.058	2114.143	2739.261	3689.791	5068.554	6369.315	7653.857	9882.696
2000	1573.030	2192.861	2619.840	3641.898	4552.136	5988.616	7988.431	9640.032	12125.054
2001	1957.227	2485.358	3159.452	4031.572	5149.951	6576.400	9495.153	11104.000	14445.570
2002	2399.965	3166.341	3452.076	4407.873	5630.470	7952.853	10495.607	12027.401	14900.667
2003	2395.094	3205.840	3618.477	4929.368	6709.939	8924.874	11324.120	13063.751	19303.770
2004	2615.740	3619.944	4059.289	5195.565	7150.291	9628.668	12126.825	13621.478	17282.914
2005	3192.201	3780.631	4188.291	5452.376	7303.525	9722.280	13299.742	14215.151	19107.678

DISCARDS FOR F-BASED PROJECTIONS

YEAR	AVG DISCARDS (000 MT)	STD

1994	0.121	0.001
1995	0.246	0.032
1996	0.245	0.111
1997	0.244	0.141
1998	0.249	0.132
1999	0.290	0.148
2000	0.328	0.175
2001	0.361	0.185
2002	0.401	0.213
2003	0.421	0.201
2004	0.442	0.268
2005	0.463	0.252

PERCENTILES OF DISCARDS (000 MT)

YEAR	1%	5%	10%	25%	50%	75%	90%	95%	99%
1994	0.119	0.120	0.120	0.120	0.121	0.121	0.121	0.122	0.122
1995	0.211	0.216	0.219	0.226	0.236	0.256	0.275	0.295	0.372
1996	0.116	0.138	0.148	0.175	0.218	0.289	0.358	0.427	0.639
1997	0.062	0.092	0.110	0.155	0.213	0.298	0.398	0.486	0.711
1998	0.066	0.091	0.117	0.153	0.219	0.317	0.394	0.481	0.641
1999	0.080	0.118	0.130	0.192	0.257	0.348	0.469	0.555	0.775
2000	0.107	0.123	0.155	0.214	0.284	0.393	0.553	0.624	0.898
2001	0.109	0.146	0.185	0.233	0.306	0.454	0.595	0.703	0.962
2002	0.101	0.154	0.183	0.261	0.358	0.479	0.673	0.794	1.116
2003	0.113	0.184	0.203	0.265	0.372	0.556	0.713	0.797	0.954
2004	0.156	0.187	0.205	0.277	0.376	0.546	0.744	0.844	1.260
2005	0.146	0.197	0.220	0.291	0.392	0.560	0.782	0.935	1.388

Example 3 (example based on AGEPRO version 1.21)

The third example is taken from a hypothetical projection analysis for cod. This example illustrates a projection with age-1 recruitment, R/SSB constraints on realized recruitment, time-varying quotas, and stochastic natural mortality. The projection begins in 1994 with a time horizon of 5 years. There are 5 simulations performed for each of 200 initial population vectors. The recruitment age is age-1 and harvest is quota-based and time-varying. Realized SSB levels are compared to a threshold and natural mortality is random. The stock-recruitment model is Beverton-Holt with lognormal error and R/SSB constraints are applied to realized recruitments. Below are the list and output files for example 3.

List of Input File for Example 3

TITLE OF THE PROJECTION RUN
cod_example3

FIRST YEAR OF THE PROJECTION RUN
1994

LENGTH OF THE TIME HORIZON
7

NUMBER OF SIMULATIONS PER INITIAL POPULATION SIZE
5

POSITIVE SEED FOR RANDOM NUMBER GENERATION
53422

AGE OF RECRUITMENT
RECRUITMENT AT AGE-1

MIXED HARVEST STRATEGY
NO MIXTURE OF FISHING MORTALITIES AND QUOTAS

DISCARDS
DISCARDS AT AGE NOT INCLUDED

TYPE OF HARVEST
HARVEST SET BY QUOTA

HARVEST STRATEGY
TIME-VARYING

TARGET F
TARGET F IS NOT INCLUDED

SURVEY CATCH-AT-AGE INDEX
NO SURVEY INDEX IS PREDICTED

SSB THRESHOLD
SSB IS COMPARED TO THRESHOLD

LANDINGS BY MARKET CATEGORY
LANDINGS ARE NOT SUMMARIZED BY CATEGORY

FRACTION OF TOTAL MORTALITY PRIOR TO SPAWNING
CONSTANT IN TIME

PARTIAL RECRUITMENT

TIME-VARYING

DISCARD FRACTION AT AGE
CONSTANT IN TIME

BOUNDED RECRUITMENT
R/SSB CONSTRAINTS ARE USED

NATURAL MORTALITY
NATURAL MORTALITY IS A RANDOM VARIABLE

INITIAL POPULATION SIZE
DISTRIBUTION OF INITIAL POPULATION SIZES

NUMBER OF AGE CLASSES
10

LOWER AND UPPER BOUND FOR RANDOM NATURAL MORTALITY
0.180000 0.220000

MEAN WEIGHTS AT AGE
0.777000 1.231000 1.965000 2.931000 4.194000 5.528000 6.956000 8.913000 10.701000 15.224000

MEAN LANDED WEIGHTS AT AGE
0.991000 1.566000 2.432000 3.530000 4.833000 6.190000 7.721000 10.058000 11.006000 15.224000

FRACTION MATURE AT AGE
0.230000 0.640000 0.910000 0.980000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000

FRACTION OF TOTAL MORTALITY BEFORE SPAWNING
0.167000

STOCHASTIC RECRUITMENT MODEL
5

BEVERTON-HOLT WITH LOGNORMAL ERROR MODEL

PARAMETERS A, B, AND RESIDUAL VARIANCE
36189.488281 96182.343750 0.426757

CONVERSION COEFFICIENTS FOR SSB AND RECRUITMENT

1000.000 1000.000

R/SSB CONSTRAINTS
ENDPOINTS OF R/SSB INTERVALS
L(HIGH) L(LOW) U(LOW) U(HIGH)
0.085964 0.107112 0.446253 0.772535

CUTOFF LEVEL OF SSB
37177000.000

DISTRIBUTION OF INITIAL POPULATION SIZE

NUMBER OF INITIAL POPULATION VECTORS
200

FILE WITH INITIAL POPULATION VECTORS
gbcd94n1.dat

CONVERSION COEFFICIENT
1000.000

SSB THRESHOLD
70000000.000

TIME-VARYING PARTIAL RECRUITMENT
PARTIAL RECRUITMENT AT AGE IN YEAR: 1
0.002700 0.334000 0.820900 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000

PARTIAL RECRUITMENT AT AGE IN YEAR: 2
0.000000 0.250000 0.750000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000

PARTIAL RECRUITMENT AT AGE IN YEAR: 3
0.000000 0.250000 0.750000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000

PARTIAL RECRUITMENT AT AGE IN YEAR: 4
0.000000 0.250000 0.750000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000

PARTIAL RECRUITMENT AT AGE IN YEAR: 5
0.000000 0.250000 0.750000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000

PARTIAL RECRUITMENT AT AGE IN YEAR: 6
0.000000 0.250000 0.750000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000

PARTIAL RECRUITMENT AT AGE IN YEAR: 7
0.000000 0.250000 0.750000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000

HARVEST SET BY QUOTA
TIME-VARYING QUOTA BY YEAR
5000000.000 5000000.000 5000000.000 10000000.000 10000000.000
10000000.000 10000000.000

Output File for Example 3

PROJECTION RUN: cod_example3
INPUT FILE: example3
OUTPUT FILE: example3.out
RECRUITMENT MODEL: 5
NUMBER OF SIMULATIONS: 5

QUOTA-BASED PROJECTIONS
TIME-VARYING QUOTA
YEAR QUOTA (000 MT)
1994 5.000
1995 5.000
1996 5.000
1997 10.000
1998 10.000
1999 10.000
2000 10.000

SPAWNING STOCK BIOMASS (THOUSAND MT)
YEAR AVG SSB (000 MT) STD
1994 32.490 5.001
1995 38.894 6.583
1996 46.777 9.164
1997 56.990 13.484
1998 67.289 19.383
1999 81.083 26.456
2000 100.256 35.458

PERCENTILES OF SPAWNING STOCK BIOMASS (000 MT)

YEAR	1%	5%	10%	25%	50%	75%	90%	95%	99%
1994	21.863	24.675	26.115	28.568	32.646	35.619	38.783	41.079	43.769
1995	24.714	28.184	30.989	33.945	38.807	42.878	48.178	50.214	54.906
1996	28.061	32.673	35.675	40.005	46.553	52.123	59.216	62.703	71.654
1997	32.234	37.859	41.328	47.041	55.601	64.901	74.592	80.787	96.295
1998	32.888	39.864	44.932	53.174	64.798	77.879	93.329	102.387	122.748
1999	33.802	44.118	50.886	61.398	78.204	96.175	117.682	127.987	159.003
2000	34.108	50.692	58.981	73.949	96.228	120.971	147.508	165.768	202.163

ANNUAL PROBABILITY THAT SSB EXCEEDS THRESHOLD: 70.00000 THOUSAND MT

YEAR	Pr(SSB > Threshold Value)
1994	0.000
1995	0.000
1996	0.013
1997	0.156
1998	0.401
1999	0.614
2000	0.800

RECRUITMENT UNITS ARE: 1000.000 FISH BIRTH

YEAR	AVG RECRUITMENT	STD
1994	8479.810	4097.519
1995	11167.114	6464.247
1996	13329.513	7491.100
1997	15758.157	8593.025
1998	17352.559	10449.576
1999	20740.891	12883.358
2000	23986.729	15431.147

PERCENTILES OF RECRUITMENT UNITS ARE: 1000.000 FISH BIRTH

YEAR	1%	5%	10%	25%	50%	75%	90%	95%	99%
1994	3299.313	3751.782	4227.998	5473.120	7788.413	10414.050	13164.822	14973.680	24230.043

1995 3582.582 4263.283 4875.409 6496.478 9377.364 13562.734 20206.299 25353.598 31714.213
1996 3866.110 4752.905 5686.021 7608.120 11397.204 17577.119 23908.340 29098.461 36232.441
1997 4149.721 5601.578 6705.627 9336.188 13790.227 20449.178 27686.070 32854.441 43419.004
1998 4551.344 6047.147 7046.784 9931.970 14542.737 22495.221 31103.816 38594.492 52006.809
1999 4778.529 7001.829 8450.063 11541.086 17323.795 25926.600 38662.137 46656.734 63886.934
2000 5180.117 8155.035 9748.943 13452.345 19905.979 29559.148 43917.047 53635.949 76669.703

REALIZED F SERIES FOR QUOTA-BASED PROJECTIONS

YEAR	AVG F	STD
1994	0.162	0.029
1995	0.143	0.027
1996	0.123	0.025
1997	0.212	0.053
1998	0.184	0.061
1999	0.157	0.069
2000	0.130	0.092

PERCENTILES OF REALIZED F SERIES

YEAR	1%	5%	10%	25%	50%	75%	90%	95%	99%
1994	0.112	0.121	0.130	0.141	0.156	0.180	0.199	0.216	0.249
1995	0.098	0.105	0.111	0.124	0.138	0.160	0.177	0.192	0.224
1996	0.078	0.088	0.093	0.105	0.118	0.137	0.153	0.169	0.200
1997	0.115	0.140	0.153	0.177	0.204	0.241	0.278	0.309	0.374
1998	0.086	0.107	0.118	0.143	0.174	0.215	0.259	0.286	0.366
1999	0.067	0.084	0.093	0.115	0.144	0.186	0.231	0.266	0.343
2000	0.052	0.065	0.072	0.090	0.115	0.152	0.192	0.229	0.339

References

Brodziak, J. and P. Rago. Unpublished manuscript 1994. A general approach for short-term stochastic projections in age-structured fisheries assessment models. Population Dynamics Branch. Northeast Fisheries Science Center. Woods Hole, Massachusetts, 02543.

Brodziak, J., P. Rago, and R. Conser. 1998. A general approach for making short-term stochastic projections from an age-structured fisheries assessment model. In F. Funk, T. Quinn II, J. Heifetz, J. Ianelli, J. Powers, J. Schweigert, P. Sullivan, and C.-I. Zhang (Eds.), *Proceedings of the International Symposium on Fishery Stock Assessment Models for the 21st Century*. Alaska Sea Grant College Program, Univ. of Alaska, Fairbanks.

Metcalf, M. 1985. Effective Fortran 77. Oxford University Press, Oxford, U.K., 231 p.

Metcalf, M., and J. Reid. 1998. Fortran 90/95 Explained. Oxford University Press, Oxford, U.K., 333 p.

Northeast Fisheries Science Center [NEFSC]. 1994. Report of the 18th Northeast Regional Stock Assessment Workshop: Stock Assessment Review Committee Consensus Summary of Assessments. NEFSC Ref. Doc. 94-22, Woods Hole, MA 02543, 199 p.

Press, W.H., S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. 1992. Numerical recipes in Fortran: The art of scientific computing, 2nd Edition. Cambridge University Press, Cambridge, U.K., 963 p.

Ricker, W.E. 1975. Computation and interpretation of biological statistics of fish populations. Bulletin of the Fisheries Research Board of Canada. Bulletin 191. Fisheries and Marine Service, Ottawa, Canada, 382 p.

Table 1. Notation for variables used in the AGEPRO model.

Variable	Description
$N_a(t)$	The number of age-a fish at the beginning of year t.
$C_a(t)$	The number of age-a fish that are captured and die in year t.
$F_a(t)$	The instantaneous fishing mortality rate applied to age-a fish in year t.
F(t)	The instantaneous fully-recruited fishing mortality rate in year t.
Fwb(t)	The instantaneous fishing mortality weighted by mean biomass in year t.
$PR_a(t)$	The partial recruitment to F(t) for age-a fish (age-specific selectivity).
M(t)	The instantaneous natural mortality rate in year t.
ZPROJ(t)	The fraction of total annual mortality that occurs from January 1st to the midpoint of the spawning season in year t.
FM_a	The average fraction of age-a fish that are mature.
$W_{S,\,a}$	The average spawning weight of an age-a fish.
SSB(t)	The total biomass of mature spawning fish measured at the midpoint of the spawning season.
MB(t)	The mean stock biomass in year t.
TSB(t)	The total stock biomass on January 1^{st} of year t.
$W_{L,\,a}$	The average weight of an age-a fish that is landed.
$W_{D,\,a}$	The average weight of an age-a fish that is discarded.
$DF_a(t)$	The fraction of age-a fish that are discarded and die in year t.
L(t)	The total weight of landed fish in year t.
D(t)	The total weight of fish that are discarded and die in year t.
Q(t)	The landings quota in year t.

Table 2. Summary of logical flags for the AGEPRO version 2.0 software ordered by their position (line number) within an input file.

Position	Logical Flag	Description
6	Age-2 Recruitment	If true, recruitment age is age-2. Otherwise recruitment age is age-1.
7	Harvest Mixture	If true, a mixture of F-based and quota-based harvest can be specified in the projection. Otherwise, harvest is either F-based or it is quota-based.
8	Discard	If true, discards at age are incorporated in the projection. Otherwise, there are no discards.
9	Quota-based	If true, catches are based on quotas. Otherwise, catches are based on fishing mortality rates.
10	Constant Harvest Strategy	If true, the harvest strategy is constant in time. Otherwise, quotas or F's can vary in time.
11	F target	If true, a target value of F is applied in any year following a year in which the SSB threshold is reached. Otherwise, F does not change after the threshold is reached.
12	Index	If true, the value of an age-specific recruitment index is predicted. Otherwise, no prediction is made.
13	SFA Threshold	If true, the realized SSB, TSB, MB, F, and Fwb are compared to their threshold. Otherwise, no comparisons are made.
14	Market Category	If true, landings are summarized by market category and output to file. Otherwise, no market category summaries are made.
15	Total Mortality	If true, the fraction of total mortality that occurs prior to spawning can vary in time. Otherwise, the fraction is constant.
16	Partial Recruitment	If true, the partial recruitment to fishing mortality at age vector can vary in time. Otherwise, the partial recruitment vector is constant.
17	Constant Discard	If true, discard proportions at age are constant. Otherwise, the discard proportion at age can vary in time.

18 Bounded Recruitment If true, realized recruitment from recruitment models with lognormal error terms is constrained based on R/SSB ratios. Otherwise, no constraints are applied to recruitment.

19 Constant Natural If true, natural mortality is constant in time. Otherwise, it varies stochastically Mortality and is a uniformly distributed random variable.

20 Bootstrap If true, a file of bootstrapped initial population vectors is used for the projection analysis. Otherwise, a single initial population vector is used.

Table 3. Structure of an input file. Values on a single line can delimited by a comma or a space.

Input #	Description	Is input required?
1	Name of projection run, input: up to 64 character string	Yes
2	First year of projection run, input: 4-digit year (Positive integer)	Yes
3	Length of planning horizon, input: Y (Positive integer)	Yes
4	Number of simulations per initial population vector, input: Positive integer	Yes
5	Number of “warmups” for random number generator, input: Positive integer	Yes
6	Age-2 recruitment flag, input: Integer (1=true; 0=false)	Yes
7	Harvest mixture flag, input: Integer (1=true; 0=false)	Yes
8	Discard flag, input: Integer (1=true; 0=false)	Yes
9	Quota-based flag, input: Integer (1=true; 0=false)	Yes
10	Constant harvest strategy flag, input: Integer (1=true; 0=false)	Yes
11	F target flag, input: Integer (1=true; 0=false)	Yes
12	Index flag, input: Integer (1=true; 0=false)	Yes
13	SFA threshold flag, input: Integer (1=true; 0=false)	Yes
14	Market category flag, input: Integer (1=true; 0=false)	Yes
15	Total mortality flag, input: Integer (1=true; 0=false)	Yes
16	Partial recruitment flag, input: Integer (1=true; 0=false)	Yes
17	Constant discard flag, input: Integer (1=true; 0=false)	Yes
18	Bounded recruitment flag, input: Integer (1=true; 0=false)	Yes
19	Constant natural mortality flag, input: Integer (1=true; 0=false)	Yes

20	Bootstrap flag, input: Integer (1=true; 0=false)	Yes
21	Number of ageclasses, lower and upper bound on range of ages for computing mean biomass, input: A, Lowerage,Upperage	Yes
22	Natural mortality rate form depends on input If input #19=true, then input: M#19 and input #6 If input #19=false and input #6=false, then input: L_M , U_M If input #19=false and input #6=true, then input: L_M , U_M and on the next line input: M(0)	Yes;
23	Mean spawning weights at age, input: $W_{S,1}$, $W_{S,2}$,$W_{S,3}$,... , $W_{S,A}$	Yes
24	Mean landed weights at age, input: $W_{L,1}$, $W_{L,2}$,$W_{L,3}$,... , $W_{L,A}$	Yes
25	Mean discarded weights at age, input: $W_{D,1}$, $W_{D,2}$,$W_{D,3}$,... , $W_{D,A}$ No; required if input #8= true	
26	Fraction mature at age, input: FM_1 , FM_2 , FM_3 ,... , FM_A	Yes

Table 3. Continued.

Input #	Description	Is input required?
27	Fraction of total mortality that occurs before spawning	Yes; depends on input #15
28	If input #15=false, then input: ZPROJ and input #6 If input #15=true and input #6=false, input: ZPROJ(1) , ZPROJ(2) , ..., ZPROJ(Y) If input #15=true and input #6=true, input: ZPROJ(0) and on the next line input: ZPROJ(1) , ZPROJ(2) , ..., ZPROJ(Y)	
28	Recruitment flag, input: Integer (between 1 and 15)	Yes
29	Recruitment model parameters depends on input #28 If input #28=1, input: K and on the next line input: $N_{R,1}$, $N_{R,2}$, $N_{R,3}$, ..., $N_{R,K}$ and on the next line input: J and on the next line input: SSB_2 , SSB_3 , SSB_4 , ..., SSB_J and on the next J lines input: $p_{1,1}$, $p_{1,2}$, $p_{1,3}$, ..., $p_{1,K}$ $p_{2,1}$, $p_{2,2}$, $p_{2,3}$, ..., $p_{2,K}$	Yes;

...

$p_{J,1}$, $p_{J,2}$, $p_{J,3}$, ..., $p_{J,K}$

If input #28=2, input: T

and on the next line input: $N_R(1)$, $N_R(2)$, $N_R(3)$, ..., $N_R(T)$

and on the next line input: SSB(1-R) , SSB(2-R) , SSB(3-R) , ..., SSB(T-R)

If input #28=3, input: T

and on the next line input: $N_R(1)$, $N_R(2)$, $N_R(3)$, ..., $N_R(T)$

If input #28=4, input: T_{LOW} , T_{HIGH}

and on the next line input: SSB*

and on the next line the LOW-SSB STATE RECRUITMENTS: $N_R(1)$, $N_R(2)$, $N_R(3)$, ..., $N_R(T_{LOW})$

and on the next line the LOW-SSB STATE SSBs: SSB(1-R) , SSB(2-R) , SSB(3-R) , ..., SSB(T_{LOW}-R)

and on the next line the HIGH-SSB STATE RECRUITMENTS: $N_R(1)$, $N_R(2)$, $N_R(3)$, ..., $N_R(T_{HIGH})$

and on the next line the HIGH-SSB STATE SSBs: SSB(1-R) , SSB(2-R) , SSB(3-R) , ..., SSB(T_{HIGH}-R)

If input #28=5, input: a , b , σ_w^2

and on the next line input: c_{SSB} , c_R

If input #28=6, input: a , b , σ_w^2

and on the next line input: c_{SSB} , c_R

If input #28=7, input: a , b, k , σ_w^2

and on the next line input: c_{SSB} , c_R

If input #28=8, input: μ_{logR} and σ_{logR}

and on the next line input: c_{SSB} , c_R

If input #28=9, input: T

and on the next line input: $N_R(1,1)$, $N_R(1,2)$, $N_R(1,3)$, ..., $N_R(1,T)$

and on the next line input: $N_R(2,1)$, $N_R(2,2)$, $N_R(2,3)$, ..., $N_R(2,T)$

...

and on the next line input: $N_R(Y,1)$, $N_R(Y,2)$, $N_R(Y,3)$, ..., $N_R(Y,T)$

If input #28=10, input: a , b , σ_w^2

and on the next line input: φ , last_log-scale_recruitment residual from stock-recruitment curve fit

and on the next line input: c_{SSB} , c_R

If input #28=11, input: a , b , σ_w^2

and on the next line input: φ , last_log-scale_recruitment residual from stock-recruitment curve fit

and on the next line input: c_{SSB} , c_R

If input #28=12, input: a , b , k, σ_w^2

and on the next line input: φ , last_log-scale_recruitment residual from stock-recruitment curve fit

and on the next line input: c_{SSB} , c_R

If input #28=13, input: μ_{logR} and σ_{logR}

and on the next line input: φ , last_log-scale_recruitment residual from stock-

recruitment curve fit
and on the next line input: c_{SSB} , c_R
If input #28=14, input: T
and on the next line input: $N_R(1)$, $N_R(2)$, $N_R(3)$, ..., $N_R(T)$
If input #28=15, input: T_{LOW} , T_{HIGH}
and on the next line input: SSB*
and on the next line the LOW-SSB STATE RECRUITMENTS: $N_R(1)$, $N_R(2)$, $N_R(3)$, ..., $N_R(T_{LOW})$
and on the next line the HIGH-SSB STATE RECRUITMENTS: $N_R(1)$, $N_R(2)$, $N_R(3)$, ..., $N_R(T_{HIGH})$

Table 3. Continued

Input #	Description	Is input required?
30	R/SSB constraints, input: L_{HIGH} , L_{LOW} , U_{LOW} , U_{HIGH} No; required if input #18=true and on the next line input: SSB_{CUT} and input #28 = 5, 6, 7, or 8	
31	Initial population abundance parameters If input #6=false and input #20=true, input: B and on the next line input: NAME OF FILE WITH A TOTAL OF B BOOTSTRAPPED **n**(1)s and on the next line input: c_N If input #6=false and input #20=false, input: c_N and on the next line input: $n_1(t)$, $n_2(t)$, $n_3(t)$, ..., $n_A(t)$ If input #6=true and input #20=true, input: B and on the next line input: NAME OF FILE WITH A TOTAL OF B BOOTSTRAPPED **n**(1)s and on the next line input: c_N and on the next line input: NAME OF FILE WITH A TOTAL OF B BOOTSTRAPPED F(0)s If input #6=true and input #20=false, input: c_N and on the next line input: $n_1(t)$, $n_2(t)$, $n_3(t)$, ..., $n_A(t)$ and on the next line input: F(0)	Yes; depends on input #6 and input #20
32	Survey catch-at-age index parameters, input: p and on the next line input: s_p* and on the next line input: β_0 , β_1	No; required if input #12=true
33	SFA thresholds, input: $SSB_{THRESHOLD}$,$TSB_{THRESHOLD}$, $F_{THRESHOLD}$, $MB_{THRESHOLD}$, $Fwb_{THRESHOLD}$	No;

required if input #13=true

34 F target parameters, input: F_{TARGET} No; required if input #11=true
and on the next line input: Y_{TARGET}

35 Fishery selectivity parameters Yes; depends on input #6 and
If input #6=true and input #16=true, input: $PR_2(0)$, $PR_3(0)$, ... , $PR_A(0)$
input #16
and on the next Y lines input: $PR_2(1)$, $PR_3(1)$, ... , $PR_A(1)$
$PR_2(2)$, $PR_3(2)$, ... , $PR_A(2)$
...
$PR_2(Y)$, $PR_3(Y)$, ... , $PR_A(Y)$
If input #6=false and input #16=true, on the next Y lines input: $PR_1(1)$, $PR_2(1)$, ... , $PR_A(1)$
$PR_1(2)$, $PR_2(2)$, ... , $PR_A(2)$
...
$PR_1(Y)$, $PR_2(Y)$, ... , $PR_A(Y)$
If input #6=true and input #16=false, input: PR_2 , PR_3 , ... , PR_A
If input #6=false and input #16=false, input: PR_1 , PR_2 , ... , PR_A

36 Discard parameters No; required if input #8=true
If input #17=true, input: DF_1 , DF_2 , ... , DF_A
and depends on input #17
If input #17=false, on the next Y lines input: $DF_1(1)$,$DF_2(1)$, ... , $DF_A(1)$
$DF_1(2)$, $DF_2(2)$, ... , $DF_A(2)$
...
$DF_1(Y)$, $DF_2(Y)$, ... , $DF_A(Y)$

37 Harvest strategy parameters Yes; depends on input #7,
If input #7=false, input #9=true, and input #10=true, input: Q
input #9, and input #10
If input #7=false, input #9=true, and input #10=false, input: Q(1) , Q(2) , Q(3) , ... , Q(Y)
If input #7=false, input #9=false, and input #10=true, input: F
If input #7=false, input #9=false, and input #10=false, input: F(1) , F(2) , F(3) , ... , F(Y)
If input #7=true, input: I(1) , I(2) , I(3) , ... , I(Y)
where I(year)=1 for a quota-based harvest and I(year)=0 for an F-based harvest in a given year
and on the next line input: Q(1) , Q(2) , Q(3) , ... , Q(Y) with dummy values for F-

based years

and on the next line input: F(1) , F(2) , F(3) , ... , F(Y) with dummy values for quota-based years

Table 3. Continued.

Input #	Description	Is input required?
38	Market category parameters, input: MC (integer between 1 and 3) No; required if input #14=true and on the next MC lines input: $q_{1,1}$, $q_{2,1}$, $q_{3,1}$, ..., $q_{A,1}$ $q_{1,2}$, $q_{2,2}$, $q_{3,2}$, ..., $q_{A,2}$ (If necessary) $q_{1,3}$, $q_{2,3}$, $q_{3,3}$, ..., $q_{A,3}$ (If necessary) and on the next line input: NAME OF OUTPUT FILE FOR MARKET CATEGORY DATA	

Appendix 1.

Application of Newton's method to solve for F

Let g(F) = L(F) - Q where L(F) is defined in Eq. 10. The first order Taylor series expansion of g(F) about a real number $x \in [0,\infty)$ results in

$$g(F) = g(x) + g\mathrm{N}(x)(F\_x)$$

Since g(F) = 0, we obtain

$$F = x - \frac{g(x)}{g\mathrm{N}(x)}$$

The iterative solution for F can be obtained by successively substituting values of F into Equation 40 such that

$$F^{(n+1)} = F^{(n)} - \frac{g(F^{(n)})}{g\mathrm{N}(F^{(n)})}$$

The function g'(F) is the first derivative of L(F) - Q with respect to F and can be expressed as g'(F) = L'(F) - 0 or

$$L\mathrm{N}(F) = \sum_{a=1}^{A} (1\_DF_a)\, W_{L,a}\, C_a^{\mathrm{N}}(F)$$

The derivative of catch with respect to F can be derived by taking the derivative of F with respect to C in Equation 5. After some algebra the derivative g'(F) reduces to

$$g\mathrm{N}(F) = \frac{\sum_{a=1}^{A} (1 - DF_a)\, W_{L,a} \dfrac{PR_a N_a}{(M + PR_a F)^2}\, \mathrm{C}}{x \left[M + (M\,\mathrm{C}PR_a F - M + PR_a^2 F^2)\, e^{-M - PR_a F} \right]}$$

$$F^{(n+1)} = F^{(n)} - \frac{L(F^{(n)}) - Q}{g\mathrm{N}(F^{(n)})}$$

Therefore, the iterative solution for F that results in catch of the quota Q can be found from The subroutine *rtsafe()* of Press *et al.* (1992) is used to ensure convergence of the iterates $F^{(n)}$ to a solution of g(F)=0. This subroutine forces the iterates $F^{(n)}$ to remain within a prescribed interval when a Newton iteration would fall outside an interval of feasible solutions. In this application the interval of feasible solutions for g(F)=0 is [0, 25] and iteration stops when $| F^{(n+1)} - F^{(n)} | < 0.0005$.

Definition of Infeasible Quotas

An infeasible quota Q occurs when the desired catch cannot be removed from the population for the maximum feasible level of fishing mortality, denoted by F^*. In theory the maximum level of F could be infinite, however, we choose to use a realistic upper bound on F^* rather than to assume that infinite fishing mortality could be applied to a commercially exploited marine fish stock. In particular, we assume the maximum feasible level of F is $F^*=25.0$. Given this choice of F^* and assuming that M=0.2, it follows that the survival probability of a recruited animal would be $e^{-Z} = e^{-25.2} \approx 1.13705 \cdot 10^{-11}$, or roughly 1 chance in 100 billion. This survival probability was considered to be small enough to characterize the possible effects of fishing

$$L^*(F^*) = \sum_{a=1}^{A} \left[1 - DF_a(t) \right] W_{L,a} \frac{PR_a(t) F^*}{M(t) + PR_a(t) F^*} \left\{ 1 - e^{-[M(t) + PR_a(t) F^*]} \right\} N_a(t)$$

mortality on a stock. The maximum landings in time period t, denoted by L^*, are then given by

Appendix 2.

Source code for agepro_v2.02.f90

```
    module global_arrays

!c      REVISED JUL-2002
            integer*4 nage
            integer*4 maxboot, maxsim, maxtime, maxlen
            integer*4 ttime

            logical discflag, allfeasible

            real*8 m, quota, tmpF
            real*8 p_feasible
            real*8, allocatable :: n(:),pr(:),wtland(:),discfrac(:)

!c      REVISED FEB-2002
            parameter(maxobsrec=100,maxpct=9, maxage=50, maxmc=3)

            integer*4,allocatable :: mixyr(:)
            logical,allocatable :: feasible(:,:)

            real*8 catch(1:maxage)

            real*8, allocatable :: var_pr(:,:),maxcatch(:)
            real*8, allocatable :: var_discfrac(:,:)

            real*8, allocatable :: resid(:)
            real*8, allocatable :: var_zproj(:)
            real*8, allocatable :: boot_n(:,:)
            real*8, allocatable :: boot_f(:)

            real*8, allocatable :: fseries(:),qseries(:)
            real*8, allocatable :: simssb(:,:,:)

!c      REVISED 7/1/99
            real*8, allocatable :: simmeanB(:,:,:)

!c      REVISED FEB-2002
            real*8, allocatable :: simtotB(:,:,:)
            real*8, allocatable :: simland(:,:,:)
            real*8, allocatable :: simdisc(:,:,:)
            real*8, allocatable :: simf(:,:,:)
            real*8, allocatable :: simrecr(:,:,:)
```

```
                real*8, allocatable :: simsvind(:,:,:)
                real*8, allocatable :: avgssb(:),avgland(:),avgsvind(:)
                real*8, allocatable :: avgdisc(:),avgf(:),avgrecr(:)

!c      REVISED 7/1/99
          real*8, allocatable :: avgmeanB(:),sdmeanB(:)
!c      REVISED FEB-2002
          real*8, allocatable :: avgtotB(:),sdtotB(:)
                real*8, allocatable :: sdssb(:),sdland(:),sdsvind(:)
                real*8, allocatable :: sddisc(:),sdf(:),sdrecr(:)
                real*8, allocatable :: pctssb(:,:),pctdisc(:,:)

!c      REVISED 7/1/99
                real*8, allocatable :: pctmeanB(:,:)
                real*8, allocatable :: pcttotB(:,:)
                real*8, allocatable :: pctland(:,:),pctf(:,:)
                real*8, allocatable :: pctrecr(:,:),pctsvind(:,:)
                real*8, allocatable :: crit_count(:),p_index(:)
                real*8, allocatable :: ssb_count(:),p_ssbthresh(:)

!c      REVISED JUL-2002
          real*8, allocatable :: pjoint_ssbthresh(:)
          real*8, allocatable :: pjoint_meanBthresh(:)
          real*8, allocatable :: pjoint_totBthresh(:)
          real*8, allocatable :: pjoint_FmeanBthresh(:)
          real*8, allocatable :: pjoint_Fthresh(:)
          real*8, allocatable :: pjoint_index(:)

!c      REVISED 7/8/99
                real*8, allocatable :: meanB_count(:),p_meanBthresh(:)
                real*8, allocatable :: FmeanB_count(:),p_FmeanBthresh(:)

!c      REVISED FEB-2002
                real*8, allocatable :: totB_count(:),p_totBthresh(:)
                real*8, allocatable :: F_count(:),p_Fthresh(:)

          real*8, allocatable :: simFmeanB(:,:,:),avgFmeanB(:)
          real*8, allocatable :: sdFmeanB(:),pctFmeanB(:,:)
                real*8, allocatable :: market(:,:)
             real*8, allocatable :: simmc1_w(:,:,:)
                real*8, allocatable :: simmc1_n(:,:,:)
                real*8, allocatable :: simmc2_w(:,:,:)
                real*8, allocatable :: simmc2_n(:,:,:)
                real*8, allocatable :: simmc3_w(:,:,:)
```

```
                real*8, allocatable :: simmc3_n(:,:,:)
                real*8, allocatable :: avgmc1_w(:)
                real*8, allocatable :: avgmc1_n(:)
                real*8, allocatable :: avgmc2_w(:)
                real*8, allocatable :: avgmc2_n(:)
                real*8, allocatable :: avgmc3_w(:)
                real*8, allocatable :: avgmc3_n(:)
                real*8, allocatable :: sdmc1_w(:)
                real*8, allocatable :: sdmc1_n(:)
                real*8, allocatable :: sdmc2_w(:)
                real*8, allocatable :: sdmc2_n(:)
                real*8, allocatable :: sdmc3_w(:)
                real*8, allocatable :: sdmc3_n(:)
                real*8, allocatable :: pctmc1_w(:,:)
                real*8, allocatable :: pctmc1_n(:,:)
                real*8, allocatable :: pctmc2_w(:,:)
                real*8, allocatable :: pctmc2_n(:,:)
                real*8, allocatable :: pctmc3_w(:,:)
                real*8, allocatable :: pctmc3_n(:,:)
                real*8, allocatable :: obsrec3(:,:)

        end


          program agepro_v2_02
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!!c   agepro_v2.02.f90 performs medium-term projections
!!c   for an exploited, age-structured population
!!c   by jon brodziak
!!c   nmfs/nefsc/woods hole laboratory
!!c   version 2.02 23-JUL-2002
!!c   double precision and F90 conversion by laura shulman
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!!c        VARIABLE DECLARATIONS
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

    use global_arrays

    real*8 ftol,fmin,fmax
    parameter (maxrec=20, maxssb=10)
    parameter (ftol=0.0005, fmin=0.0000, fmax=25.0)
```

```
!c    REVISED 6/5/02
      character*64 runname
      character*128 infile,bfile1,bfile2,outfile,mcfile

!c       REVISED 7/8/99 NO EXTERNAL FILES CREATED WITHOUT A FLAG
!c    character*128 SSBoutfile,Foutfile,Landoutfile,meanBoutfile

      integer*4 j,jj,time,ssbindex,index,iflag,sim,boot
      integer*4 k,nsim,ntime,nrec,nssb,nboot,baseyr
      integer*4 nfeasible,recflag,nrec1,nrec2,nmc,index_age
      integer*4 year,ftaryear

!c       REVISED 7/1/99
      integer*4 lowerage,upperage
      integer option

      logical age2recflag,existflag
      logical done,quotaflag,constflag
      logical indexflag,sfaflag
      logical mcflag,mixflag,use_ftarget,ftarflag
      logical zprojflag,prflag,constdiscflag
      logical bdrecflag,constmflag,bootflag

      real*8 p_reclevel(1:maxssb,1:maxrec),recruit(1:maxrec)
      real*8 past_n(1:maxage)
      real*8 next_n(1:maxage)

      real*8 uniform, reclevel, zproj, bootunit
      real*8 m_upper,m_lower,ssb_value
      real*8 past_zproj,past_m
      real*8 ssb_cut(maxssb-1)
      real*8 past_ssb

!c       REVISED 7/1/99
      real*8 ubland,fquota,landings,discards,ssb,h,gamma,meanB

!c       REVISED 7/8/99
!c       REVISED FEB-2002
      real*8 f,FmeanB, totB

      real*8 obsrec(1:maxobsrec),obsssb(1:maxobsrec)
      real*8 rssb(1:maxobsrec),sortrssb(1:maxobsrec)
      real*8 obsrec1(1:maxobsrec),obsrec2(1:maxobsrec)
```

```
      real*8 obsssb1(1:maxobsrec),obsssb2(1:maxobsrec)
      real*8 sort1(1:maxobsrec),sort2(1:maxobsrec)

!c        REVISED FEB-2002
      real*8 sortrec(1:maxobsrec)
      real*8 wt(1:maxage),wtdisc(1:maxage)
      real*8 fm(1:maxage)

!c        REVISED 7/8/99
!c     real*8 pctvalue(1:maxpct),ssbthresh,meanBthresh,FmeanBthresh
!c        REVISED 6/5/02
      real*8 pctvalue(1:maxpct)
      real*8 ssbthresh,meanBthresh,FmeanBthresh


!c        REVISED FEB-2002
      real*8 Fthresh,totBthresh

      real*8 ssbcut_m4
      real*8 mc_wt,mc_num

!c        REVISED FEB-2002
      real*8 residvar,ssb_srr
      real*8 phi,sigmasqw,last_resid
      real*8 a_bhsrr,b_bhsrr

      real*8 a_ricsrr,b_ricsrr
      real*8 a_shsrr,b_shsrr,k_shsrr
      real*8 lnrec_mean,lnrec_std
      real*8 ssb_unit,rec_unit
      real*8 ftarget,ssb_min
      real*8 rssb_lower,rssb_upper,rssb_max,rssb_min
      real*8 past_pr(1:maxage)
      real*8 gasdev
      real*8 rtsafe,ran2

!c        REVISED 6/5/02
      common /params/ nfeasible,iflag,nboot,nsim,ntime
      common /pcxval/ pctvalue

      allocate(n(maxage),pr(maxage),wtland(maxage),discfrac(maxage))

!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!!c          MAIN PROGRAM
```

```
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!!c          READ SYSTEM DATA
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!!c   read input filename from standard input
!c        REVISED 6/5/02
      print *,' '
      print *,'Enter the input filename:'
      read (*,'(a128)') infile

!c   read output filename from standard input
!c        REVISED 6/5/02
      print *,' '
      print *, 'Enter the output filename:'
      read (*,'(a128)') outfile

!c        REVISED 7/8/99 NO EXTERNAL FILES CREATED WITHOUT A FLAG
!c        REVISED 6/5/02
!c      print *,' '
!c      print *, 'Enter the SSB output filename:'
!c      read (*,'(a128)') SSBoutfile
!c      print *,' '
!c      print *, 'Enter the MEAN B output filename:'
!c      read (*,'(a128)') meanBoutfile
!c      print *,' '
!c      print *, 'Enter the F output filename:'
!c      read (*,'(a128)') Foutfile
!c      print *,' '
!c      print *, 'Enter the Landings output filename:'
!c      read (*,'(a128)') Landoutfile


!c   check whether input file exists
      inquire(file=infile,exist=existflag)

!c   exit if input file does not exist
      if (existflag .eqv. .false.) then
        print *,' '
        print *,'Input file does not exist.'
        print *,' '
        print *,'Exiting projection analysis.'
        print *,' '
        goto 99999
```

```
      endif

!c   check whether output file exists
      inquire(file=outfile,exist=existflag)

!c   exit if output file already exists
      if (existflag .eqv. .true.) then
        print *,' '
        print *,'Output file already exists.'
        print *,' '
        print *,'Exiting projection analysis.'
        print *,' '
        goto 99999
      endif

!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!c          READ INPUT DATA
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!c   read input data from file named "infile"
      open(unit=1,file=infile,status='unknown')

!c   READ RUN DESCRIPTOR FROM INFILE
!c   read runname, a descriptor for this computation
      read(1,'(a64)') runname

!c   READ SIMULATION DATA
!c   read base year for time horizon
      read(1,*) baseyr

!c   read length of time horizon
      read(1,*) ntime
            maxtime=ntime

!c   read number of simulations to perform
      read(1,*) nsim
            maxsim=nsim

!c   read number of reps to initialize the random number generator
      read(1,*) nreps

!c   read the age2recflag
!c   iflag=1 ==> age of recruitment is age-2
!c   otherwise age of recruitment is age-1
      read(1,*) iflag
```

```
      if (iflag .eq. 1) then
        age2recflag=.true.
      else
        age2recflag=.false.
      endif

!c    read the mixflag
!c    iflag=1 ==> catch projections are based on either F or a QUOTA
!c    for each year in the planning horizon
      read(1,*) iflag
      if (iflag .eq. 1) then
        mixflag=.true.
      else
        mixflag=.false.
      endif

!c    read the discard flag
!c    iflag=1 ==> discarding is included in projections
!c    and discflag is set to TRUE
      read(1,*) iflag
      if (iflag .eq. 1) then
        discflag=.true.
      else
        discflag=.false.
      endif

!c    read the quotaflag
!c    iflag=1 ==> catch projections are based on quotas
!c    and quotaflag is set to TRUE. Otherwise catch projections
!c    are based upon fishing mortality rates.
      read(1,*) iflag
      if (iflag .eq. 1) then
        quotaflag=.true.
      else
        quotaflag=.false.
      endif

!c    read the constflag
!c    iflag=1 ==> F or quota level is CONSTANT
!c    and constflag is set to TRUE.
!c    Otherwise the F or quota level may VARY IN TIME
      read(1,*) iflag
      if (iflag .eq. 1) then
        constflag=.true.
```

```
      else
        constflag=.false.
      endif

!c    read the ftarflag
!c    iflag=1 ==> ftarget is applied in year ftaryear after SSB threshold reached
      read(1,*) iflag
      if (iflag .eq. 1) then
        ftarflag=.true.
      else
        ftarflag=.false.
      endif

!c    read the indexflag
!c    iflag=1 ==> compute age-specific survey index of recruitment
      read(1,*) iflag
      if (iflag .eq. 1) then
        indexflag=.true.
      else
        indexflag=.false.
      endif

!c    read the sfaflag
!c    iflag=1 ==> compare realized ssb,meanb,and f_wt_b levels to thresholds
      read(1,*) iflag
      if (iflag .eq. 1) then
        sfaflag=.true.
      else
        sfaflag=.false.
      endif

!c    read the mcflag
!c    iflag=1 ==> output market category summaries to file
      read(1,*) iflag
      if (iflag .eq. 1) then
        mcflag=.true.
      else
        mcflag=.false.
      endif

!c    read the zprojflag
!c    iflag=1 ==> zproj varies through the planning horizon
      read(1,*) iflag
      if (iflag .eq. 1) then
```

```
      zprojflag=.true.
     else
      zprojflag=.false.
     endif

!c   read the prflag
!c   iflag=1 ==> pr vector varies through the planning horizon
     read(1,*) iflag
     if (iflag .eq. 1) then
      prflag=.true.
     else
      prflag=.false.
     endif

!c   read the constdiscflag
!c   iflag=1 ==> discard fraction at age is constant
     read(1,*) iflag
     if (iflag .eq. 1) then
      constdiscflag=.true.
     else
      constdiscflag=.false.
     endif

!c   read the bdrecflag
!c   iflag=1 ==> recruitment is bounded by r/ssb constraints
!c   for stochastic Beverton-Holt, Ricker, or Shepherd SRR
     read(1,*) iflag
     if (iflag .eq. 1) then
      bdrecflag=.true.
     else
      bdrecflag=.false.
     endif

!c   read the constmflag
!c   iflag=1 ==> natural mortality is constant
!c             else use a uniform[m_lower,m_upper] rv
!c   for stochastic natural mortality by year
     read(1,*) iflag
     if (iflag .eq. 1) then
      constmflag=.true.
     else
      constmflag=.false.
     endif
```

```
!c  read the bootflag
!c  iflag=1 ==> use file of bootstrap initial
!c              population sizes at age
!c  otherwise read 1 initial population vector
!c  from the input file
      read(1,*) iflag
      if (iflag .eq. 1) then
        bootflag=.true.
      else
        bootflag=.false.
      endif

      call OtherAlloc

!c  READ BIOLOGICAL DATA
!c  read number of age classes-last age is a plus-group
!c  AND READ LOWER AND UPPER AGE FOR COMPUTING MEAN BIOMASS

!c        REVISED 7/1/99
      read(1,*) nage,lowerage,upperage

      if (constmflag .eqv. .true.) then
!c  read instantaneous natural mortality rate
        read(1,*) m
!c  set constant past value of m
        past_m=m
      else
!c  read parameters for uniform distribution of m
        read(1,*) m_lower,m_upper
!c  read previous value of m, AGE-2 RECRUITMENT
        if (age2recflag .eqv. .true.) then
          read(1,*) past_m
        endif
      endif

!c  read mean weights at age for the stock
      read(1,*) (wt(k),k=1,nage)

!c  read mean weights at age for landed fish
      read(1,*) (wtland(k),k=1,nage)

!c  read discard weights at age, IF APPLICABLE
      if (discflag .eqv. .true.) then
        read(1,*) (wtdisc(k),k=1,nage)
```

```
      endif

!c   read maturity probability at age
      read(1,*) (fm(k),k=1,nage)

!c   read fraction of total mortality that occurs before spawning
      if (zprojflag .eqv. .true.) then
!c   zproj varies by time period
!c   read past value of zproj for AGE-2 RECRUITMENT
        if (age2recflag .eqv. .true.) then
          read(1,*) past_zproj
        endif
!c   read the value of zproj by time period
        read(1,*) (var_zproj(k), k=1,ntime)
      else
!c   read the constant value of zproj
        read(1,*) zproj
!c   set past value of zproj
        past_zproj=zproj
      endif

!c   read recflag
!c   recflag=1 ==> MODEL 1 (MARKOV MATRIX)
!c   recflag=2 ==> MODEL 2 (NONPARAMETRIC DENSITY-INDEPENDENT SRR)
!c   recflag=3 ==> MODEL 3 (EMPIRICAL RECRUITMENT)
!c   recflag=4 ==> MODEL 4 (2-SSB STATE NONPARAMETRIC SRR)
!c   recflag=5 ==> MODEL 5 (BEVERTON-HOLT WITH LOGNORMAL ERROR)
!c   recflag=6 ==> MODEL 6 (RICKER WITH LOGNORMAL ERROR)
!c   recflag=7 ==> MODEL 7 (SHEPHERD WITH LOGNORMAL ERROR)
!c   recflag=8 ==> MODEL 8 (LOGNORMAL DISTRIBUTION)
!c   recflag=9 ==> MODEL 9 (TIME-VARYING EMPIRICAL RECRUITMENT)
!c   recflag=10 ==> MODEL 10 (BEVERTON-HOLT WITH LOGNORMAL AR(1) ERROR)
!c   recflag=11 ==> MODEL 11 (RICKER WITH LOGNORMAL AR(1) ERROR)
!c   recflag=12 ==> MODEL 12 (SHEPHERD WITH LOGNORMAL AR(1) ERROR)
!c   recflag=13 ==> MODEL 13 (LOGNORMAL DISTRIBUTION WITH AR(1) ERROR)
!c   recflag=14 ==> MODEL 14 (EMPIRICAL CDF OF RECRUITMENT)
!c   recflag=15 ==> MODEL 15 (TWO-STAGE EMPIRICAL CDF OF RECRUITMENT)
      read(1,*) recflag

!c   read data for the chosen recruitment model
      if (recflag .eq. 1) then
!c   MODEL 1
!c   read number of recruitment levels
        read(1,*) nrec
```

```
!c   read recruitment levels
       read(1,*) (recruit(k),k=1,nrec)
!c   read number of spawning stock levels
       read(1,*) nssb
!c   read ssb cut points to define spawning stock levels
       read(1,*) (ssb_cut(k),k=1,(nssb-1))
!c   read probability of recruitment level (k) given ssb level (j)
       do 5 j=1,nssb
         read(1,*) (p_reclevel(j,k),k=1,nrec)
5      continue

     else if (recflag .eq. 2) then
!c   MODEL 2
!c   read number of observed recruitment/ssb data points
       read(1,*) nrec
!c   read observed recruitment series
       read(1,*) (obsrec(j),j=1,nrec)
!c   read observed ssb series
       read(1,*) (obsssb(j),j=1,nrec)

     else if (recflag .eq. 3) then
!c   MODEL 3
!c   read number of observed recruitments
       read(1,*) nrec
!c   read observed recruitment series
       read(1,*) (obsrec(j),j=1,nrec)

     else if (recflag .eq. 4) then
!c   MODEL 4
!c   read the number of low (1) and high (2) SSB data points
       read(1,*) nrec1,nrec2
!c   read the cut point between low and high ssb states
       read(1,*) ssbcut_m4
!c   read the observed recruitment series for low SSB
       read(1,*) (obsrec1(j),j=1,nrec1)
!c   read the observed ssb series for low SSB
       read(1,*) (obsssb1(j),j=1,nrec1)
!c   read the observed recruitment series for high SSB
       read(1,*) (obsrec2(j),j=1,nrec2)
!c   read the observed ssb series for high SSB
       read(1,*) (obsssb2(j),j=1,nrec2)

     else if (recflag .eq. 5) then
!c   MODEL 5
```

```
!c   read the a and b parameters and residual variance
       read(1,*) a_bhsrr,b_bhsrr,residvar
!c   read the units of ssb input and recruitment output for the SRR
       read(1,*) ssb_unit,rec_unit

     else if (recflag .eq. 6) then
!c   MODEL 6
!c   read the a and b parameters and residual variance
       read(1,*) a_ricsrr,b_ricsrr,residvar
!c   read the units of ssb input and recruitment output for the SRR
       read(1,*) ssb_unit,rec_unit

     else if (recflag .eq. 7) then
!c   MODEL 7
!c   read the a, b, and k parameters and residual variance
       read(1,*) a_shsrr,b_shsrr,k_shsrr,residvar
!c   read the units of ssb input and recruitment output for the SRR
       read(1,*) ssb_unit,rec_unit

     else if (recflag .eq. 8) then
!c   MODEL 8
!c   read the parameters of the lognormal distribution
!c   log(mean) and log(std)
       read(1,*) lnrec_mean,lnrec_std
!c   read the units of recruitment output
       read(1,*) ssb_unit,rec_unit

     else if (recflag .eq. 9) then
!c   MODEL 9
!c   read the distribution of observed recruitments
!c   for each year in the time horizon
       read(1,*) nrec
       do 301 ttime=1,ntime
        read(1,*)(obsrec3(ttime,j),j=1,nrec)
301     continue

!c        REVISED FEB-2002
     else if (recflag .eq. 10) then
!c   MODEL 10
!c   read the a and b parameters and residual variance
       read(1,*) a_bhsrr,b_bhsrr,residvar
!c   read the phi parameter and the last recruitment residual
       read(1,*) phi,last_resid
!c   read the units of ssb input and recruitment output for the SRR
```

```
      read(1,*) ssb_unit,rec_unit

     else if (recflag .eq. 11) then
!c   MODEL 11
!c   read the a and b parameters and residual variance
      read(1,*) a_ricsrr,b_ricsrr,residvar
!c   read the phi parameter and the last recruitment residual
      read(1,*) phi,last_resid
!c   read the units of ssb input and recruitment output for the SRR
      read(1,*) ssb_unit,rec_unit

     else if (recflag .eq. 12) then
!c   MODEL 12
!c   read the a, b, and k parameters and residual variance
      read(1,*) a_shsrr,b_shsrr,k_shsrr,residvar
!c   read the phi parameter and the last recruitment residual
      read(1,*) phi,last_resid
!c   read the units of ssb input and recruitment output for the SRR
      read(1,*) ssb_unit,rec_unit

     else if (recflag .eq. 13) then
!c   MODEL 13
!c   read the parameters of the lognormal distribution
!c   log(mean) and log(std)
      read(1,*) lnrec_mean,lnrec_std
!c   read the phi parameter and the last recruitment residual
      read(1,*) phi,last_resid
!c   read the units of recruitment output
      read(1,*) ssb_unit,rec_unit

     else if (recflag .eq. 14) then
!c   MODEL 14
!c   read number of observed recruitment data points
      read(1,*) nrec
!c   read observed recruitment series
      read(1,*) (obsrec(j),j=1,nrec)

     else if (recflag .eq. 15) then
!c   MODEL 15
!c   read the number of low (1) and high (2) SSB data points
      read(1,*) nrec1,nrec2
!c   read the cut point between low and high ssb states
      read(1,*) ssbcut_m4
!c   read the observed recruitment series for low SSB
```

```
      read(1,*) (obsrec1(j),j=1,nrec1)
!c   read the observed recruitment series for high SSB
      read(1,*) (obsrec2(j),j=1,nrec2)
     endif

!c   read data to constrain realized r/ssb for lognormal error models
     if (bdrecflag .eqv. .true.) then
!c   read percentiles of empirical R/SSB (min,10th,90th,max)
      read(1,*) rssb_min,rssb_lower,rssb_upper,rssb_max
!c   read minimum empirical SSB (in KG)
      read(1,*) ssb_min
     endif

!c   RECRUITMENT AT AGE-1
     if (age2recflag .eqv. .false.) then
!c   read initial population vectors from a file, IF APPLICABLE
      if (bootflag .eqv. .true.) then

!c   read number of bootstrapped initial population vectors
       read(1,*) nboot

         maxboot=nboot

      call allocation

!c       REVISED 6/5/02
!c   read the name of the file containing the bootstrap data
       read(1,'(a128)') bfile1

!c   read the factor to convert the bootstrap data to
!c   absolute numbers (bootunit)
       read(1,*) bootunit

!c   read bootstrap data-each line in bfile1 corresponds
!c   to one initial population vector ordered as n(1) n(2) ...
       open(unit=2,file=bfile1,status='old')
       do 50 j=1,nboot
        read(2,*) (boot_n(j,k),k=1,nage)
50      continue
       close(2)

!c   convert the bootstrap data to absolute numbers at age
       do 60 j=1,nboot
        do 70 k=1,nage
```

```
          boot_n(j,k)=boot_n(j,k)*bootunit
70        continue
60       continue

      else
!c  otherwise read one initial population vector from the input file
        nboot=1
           maxboot=1

!c  read the factor to convert the population vector data to
!c  absolute numbers (bootunit)
        read(1,*) bootunit

!c  read initial population vector
        read(1,*) (boot_n(1,k),k=1,nage)

!c  convert the initial population vector to absolute numbers
        do 72 k=1,nage
          boot_n(1,k)=boot_n(1,k)*bootunit
72        continue
      endif

    else
!c  RECRUITMENT AT AGE-2
!c  read initial population vectors from a file, IF APPLICABLE
      if (bootflag .eqv. .true.) then
!c  read number of bootstrapped initial population vectors
        read(1,*) nboot
           maxboot=nboot

           call allocation

!c  read the name of the file containing the bootstrap data
        read(1,'(a30)') bfile1

!c  read the factor to convert the bootstrap data to
!c  absolute numbers (bootunit)
        read(1,*) bootunit

!c  read bootstrap data-each line in bfile1 corresponds
!c  to one initial population vector ordered as n(1) n(2) ...
        open(unit=2,file=bfile1,status='old')
        do 51 j=1,nboot
          read(2,*) (boot_n(j,k),k=1,nage)
```

```
51        continue
          close(2)

!c        REVISED 6/5/02
!c   read the filename containing the bootstrap f's
          read(1,'(a128)') bfile2

!c   read the bootstrap data-the f series is stored as 1 f per line
          open(unit=2,file=bfile2,status='old')
          do 44 j=1,nboot
            read(2,*) boot_f(j)
44        continue
          close(2)

!c   convert the bootstrap data to absolute numbers at age
          do 61 j=1,nboot
            do 71 k=1,nage
              boot_n(j,k)=boot_n(j,k)*bootunit
71          continue
61        continue

        else
!c   otherwise read one initial population vector from the input file
!c   and prior f
          nboot=1
            maxboot=nboot

!c   read the factor to convert the population vector data to
!c   absolute numbers (bootunit)
          read(1,*) bootunit
!c   read initial population vector
          read(1,*) (boot_n(1,k),k=1,nage)

!c   read prior f
          read(1,*) boot_f(1)

!c   convert the initial population vector to absolute numbers
          do 74 k=1,nage
            boot_n(1,k)=boot_n(1,k)*bootunit
74        continue
        endif
      endif

!c   read parameters for age-specific recruitment index, IF APPLICABLE
```

```
      if (indexflag .eqv. .true.) then

!c   read age of recruitment index
        read(1,*) index_age

!c   read critical value of index
        read(1,*) crit_index

!c   read parameters for index prediction
!c   svindex = a_intercept + b_slope * n(index_age)
        read(1,*) a_intercept, b_slope

      endif

!c   read ssb threshold, IF APPLICABLE
      if (sfaflag .eqv. .true.) then
       read(1,*) ssbthresh, totBthresh, Fthresh, meanBthresh,FmeanBthresh
      endif

!c   read ftarget and ftaryear, IF APPLICABLE
      if (ftarflag .eqv. .true.) then
       read(1,*) ftarget
       read(1,*) ftaryear
      endif

!c   READ FISHERY DATA
!c   read selectivity (pr) at age (k)
      if (prflag .eqv. .true.) then
!c   pr varies by time period

!c   read the past pr vector for AGE-2 RECRUITMENT
       if (age2recflag .eqv. .true.) then
        read(1,*) (past_pr(k),k=1,nage)
       endif

!c   read the pr vector by time period
       do 75 j=1,ntime
        read(1,*) (var_pr(j,k), k=1,nage)
75      continue

      else
!c   read the constant pr vector
       read(1,*) (pr(k),k=1,nage)
```

```
!c  set constant past value of pr
      do 78 k=1,nage
        past_pr(k)=pr(k)
78    continue
    endif

!c  read fraction discarded at age, IF APPLICABLE
    if (discflag .eqv. .true.) then

!c    read constant discard fraction at age
      if (constdiscflag .eqv. .true.) then
        read(1,*) (discfrac(k),k=1,nage)

!c    read variable discard fraction at age
      else
        do 77 j=1,ntime
          read(1,*) (var_discfrac(j,k), k=1,nage)
77      continue
      endif
    endif

!c  IF HARVESTING IS NOT A MIXTURE OF F AND QUOTA LEVELS
    if (mixflag .eqv. .false.) then
!c  read F or quota level(s)
      if (quotaflag .eqv. .true.) then
!c  QUOTA-BASED CATCHES
        if (constflag .eqv. .true.) then
!c  FIXED QUOTA
          read(1,*) quota
        else
!c  VARIABLE QUOTA
          read(1,*) (qseries(k),k=1,ntime)
        endif

      else
!c  F-BASED CATCHES
        if (constflag .eqv. .true.) then
!c  FIXED F
          read(1,*) F
        else
!c  VARIABLE F
          read(1,*) (fseries(k),k=1,ntime)
        endif
      endif
```

```
      else
!c   HARVESTING IS A MIXTURE OF F AND QUOTA LEVELS
!c   READ INDEX FOR EACH YEAR 1==>QUOTA LEVEL AND 0==>F LEVEL
       read(1,*) (mixyr(j),j=1,ntime)

!c   READ VARIABLE QUOTA SERIES (USE -1 TO INDICATE AN F-BASED YEAR)
       read(1,*) (qseries(j),j=1,ntime)

!c   READ VARIABLE F SERIES (USE -1 TO INDICATE A QUOTA-BASED YEAR)
       read(1,*) (fseries(j),j=1,ntime)
      endif

!c   read data for market category suumaries, IF APPLICABLE
      if (mcflag .eqv. .true.) then
!c   read number of market categories (MUST BE LESS THAN maxmc)
       read(1,*) nmc

!c   read proportions at age by market category
       do 80 j=1,nmc
         read(1,*) (market(j,k),k=1,nage)
80     continue

!c     REVISED 6/5/02
!c   read output filename for market category data
       read(1,'(a128)') mcfile
      endif

      maxlen=maxboot*maxsim

!c   CLOSE INPUT FILE
      close(1)

!c   END OF DATA INPUT

!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!c          INITIALIZE VARIABLES
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!c     SET THE REFERENCE PERCENTILE VALUES FOR OUTPUT
          pctvalue(1)=0.01
          pctvalue(2)=0.05
          pctvalue(3)=0.10
          pctvalue(4)=0.25
          pctvalue(5)=0.50
```

```
        pctvalue(6)=0.75
        pctvalue(7)=0.90
        pctvalue(8)=0.95
        pctvalue(9)=0.99

!c   INITIALIZE COUNTS
     do 230 j=1,maxtime
       crit_count(j)=0.
       ssb_count(j)=0.

!c    REVISED 7/8/99
!c        REVISED FEB-2002
       meanB_count(j)=0.
       FmeanB_count(j)=0.
       totB_count(j)=0.
       F_count(j)=0.
230    continue

!c   COMPUTE sigmasqw FOR AR(1) MODELS
     if ((recflag .ge. 10) .and. (recflag .le. 13)) then
       sigmasqw=(1.0-phi*phi)*residvar
     endif

!c   COMPUTE AND SORT RECRUITMENT PER SSB FOR MODEL 2, IF APPLICABLE
     if (recflag .eq. 2) then
       do 25 j=1,nrec
         rssb(j)=obsrec(j)/obsssb(j)
25     continue
       do 35 j=1,nrec
         sortrssb(j)=rssb(j)
35     continue
       do 45 j=(nrec+1),maxobsrec
         sortrssb(j)=-1
45     continue
!c   SORTED R/SSB VALUES ARE IN LOCATIONS maxobsrec-nrec+1 to maxobsrec
       call hpsort(maxobsrec,sortrssb)
     endif

!c   COMPUTE AND SORT RECRUITMENT PER SSB FOR MODEL 4, IF APPLICABLE
     if (recflag .eq. 4) then
!c   LOW SSB STATE
       do 27 j=1,nrec1
         rssb(j)=obsrec1(j)/obsssb1(j)
27     continue
```

```
      do 37 j=1,nrec1
        sort1(j)=rssb(j)
37      continue
      do 47 j=(nrec1+1),maxobsrec
        sort1(j)=-1
47      continue
!c   SORTED R/SSB VALUES ARE IN LOCATIONS maxobsrec-nrec1+1 to maxobsrec
!c   OF sort1 FOR THE LOW SSB STATE
      call hpsort(maxobsrec,sort1)
!c   HIGH SSB STATE
      do 28 j=1,nrec2
        rssb(j)=obsrec2(j)/obsssb2(j)
28      continue
      do 38 j=1,nrec2
        sort2(j)=rssb(j)
38      continue
      do 48 j=(nrec2+1),maxobsrec
        sort2(j)=-1
48      continue
!c   SORTED R/SSB VALUES ARE IN LOCATIONS maxobsrec-nrec2+1 to maxobsrec
!c   OF sort2 FOR THE HIGH SSB STATE
      call hpsort(maxobsrec,sort2)
    endif

!c        REVISED FEB-2002
!c   COMPUTE AND SORT RECRUITMENT FOR MODEL 14, IF APPLICABLE
    if (recflag .eq. 14) then
      do 1035 j=1,nrec
        sortrec(j)=obsrec(j)
1035    continue
      do 1045 j=(nrec+1),maxobsrec
        sortrec(j)=-1
1045    continue
!c   SORTED R VALUES ARE IN LOCATIONS maxobsrec-nrec+1 to maxobsrec
      call hpsort(maxobsrec,sortrec)
    endif

!c        REVISED FEB-2002
!c   COMPUTE AND SORT RECRUITMENT FOR MODEL 15, IF APPLICABLE
    if (recflag .eq. 15) then
      do 1037 j=1,nrec1
        sort1(j)=obsrec1(j)
1037    continue
      do 1047 j=(nrec1+1),maxobsrec
```

```
        sort1(j)=-1
1047    continue
!c   SORTED R VALUES ARE IN LOCATIONS maxobsrec-nrec1+1 to maxobsrec
!c   OF sort1 FOR THE LOW SSB STATE
       call hpsort(maxobsrec,sort1)
       do 1038 j=1,nrec2
         sort2(j)=obsrec2(j)
1038    continue
       do 1048 j=(nrec2+1),maxobsrec
         sort2(j)=-1
1048    continue
!c   SORTED R VALUES ARE IN LOCATIONS maxobsrec-nrec2+1 to maxobsrec
!c   OF sort2 FOR THE HIGH SSB STATE
       call hpsort(maxobsrec,sort2)
     endif

!c   WARMUP RANDOM NUMBER GENERATOR
     call warmup(nreps)
     print *,' '
     print *,'Projection analysis is running ...'

!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!c          RUN PROJECTION
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

!c   BEGIN THE SIMULATION: LOOP OVER nboot INITIAL POPULATION VECTORS
FOR
!c      nsim SIMULATIONS AND ntime TIME STEPS


!c   LOOP OVER BOOTSTRAPPED INITIAL POPULATION VECTORS
     do 100 boot=1,nboot

!c   LOOP OVER NUMBER OF SIMULATIONS
       do 200 sim=1,nsim

!c   COPY BOOTSTRAP DATA
         do 210 j=1,nage
           n(j)=boot_n(boot,j)
210       continue

!c   RECRUITMENT AT AGE-2
!c   COMPUTE POPULATION VECTOR IN PREVIOUS YEAR GIVEN F
         if (age2recflag .eqv. .true.) then
```

```
!c   SET pr TO past_pr IF pr IS TIME-VARYING
          if (prflag .eqv. .true.) then
            do 213 k=1,nage
              pr(k)=past_pr(k)
213           continue
          endif
!c   SET m TO past_m if m IS TIME-VARYING
          if (constmflag .eqv. .false.) then
            m=past_m
          endif
!c   EQN 2
          do 212 j=1,(nage-2)
            past_n(j)=n(j+1)*exp(m+boot_f(boot)*pr(j))
212         continue
          h=exp(-m-boot_f(boot))
          gamma=h/(1.0-h)
!c   EQN 7
          past_n(nage-1)=n(nage)*exp(m+pr(nage)*boot_f(boot))/(1.0+gamma)
!c   EQN 8
          past_n(nage)=gamma*past_n(nage-1)
        endif

!c   SET use_ftarget FLAG TO FALSE FOR INITIAL TIME PERIOD
!c   NOTE: use_ftarget IS DYNAMICALLY SET FOR PERIODS 2,...,NTIME
        use_ftarget=.false.

!c   LOOP OVER TIME HORIZON
        do 300 time=1,ntime

!c  SET VARIABLES THAT VARY BY TIME PERIOD
!c  INITIALIZE RECRUITMENT RESIDUAL VECTOR FOR AR(1) MODELS
          if ((recflag .ge. 10) .and. (recflag .le. 13)) then
               resid(0)=last_resid
               do 207 k=1,ntime
             resid(k)=0.0
207            continue
             endif

!c     IF zproj VARIES THROUGH TIME SET zproj=var_zproj(time)
          if (zprojflag .eqv. .true.) then
            zproj=var_zproj(time)
          endif

!c     IF pr vector VARIES THROUGH TIME SET pr=var_pr(time)
```

```
          if (prflag .eqv. .true.) then
            do 215 k=1,nage
              pr(k)=var_pr(time,k)
215           continue
          endif

!c     IF discfrac vector VARIES THROUGH TIME SET discfrac=var_discfrac(time)
          if (constdiscflag .eqv. .false.) then
            do 205 k=1,nage
              discfrac(k)=var_discfrac(time,k)
205           continue
          endif

!c     IF QUOTA LEVEL VARIES THROUGH TIME SET quota=qseries(time)
          if (mixflag .eqv. .true.) then
            quota=qseries(time)
          endif
          if ((quotaflag .eqv. .true.) .and.(constflag .eqv. .false.)) then
            quota=qseries(time)
          endif

!c   GENERATE STOCHASTIC NATURAL MORTALITY, IF APPLICABLE
          if (constmflag .eqv. .false.) then
            uniform=ran2(idum)
            m=m_lower+uniform*(m_upper-m_lower)
          endif

!c   COMPUTE CATCH AT AGE
!c   PURE F OR QUOTA-BASED CATCHES
          if (mixflag .eqv. .false.) then
            if (quotaflag .eqv. .true.) then

!c   BASED ON A PREDETERMINED QUOTA
!c   COMPUTE AN UPPER BOUND ON LANDINGS (SET F=fmax)
              call calc_catch(n,nage,maxage,fmax,pr,m,maxcatch)
              ubland=0.
!c   INCLUDE DISCARDS, IF APPLICABLE
              if (discflag .eqv. .true.) then
                do 305 j=1,nage
                  ubland=ubland+maxcatch(j)*(1.-discfrac(j))*wtland(j)
305               continue
              else
                do 310 j=1,nage
                  ubland=ubland+maxcatch(j)*wtland(j)
```

```
310             continue
              endif
!c   DETERMINE WHETHER THE QUOTA IS FEASIBLE
               if (quota .gt. ubland) then
                 feasible(boot,sim)=.false.
               else
                 feasible(boot,sim)=.true.
               endif

!c   IF QUOTA IS NOT FEASIBLE, EXIT THIS SIMULATION
              if (feasible(boot,sim) .eqv. .false.) goto 999

!c   COMPUTE F THAT TAKES THE QUOTA
              fquota=rtsafe(fmin,fmax,ftol)

!c   STORE THE REALIZED F
              simf(boot,sim,time)=fquota

!c   APPLY THIS F TO CALCULATE CATCH
              call calc_catch(n,nage,maxage,fquota,pr,m,catch)

            else
!c   ELSE COMPUTE CATCH BASED ON A PREDETERMINED F
!c   ALL F VALUES ARE FEASIBLE
          do 220 j=1, nboot
            do 225 jj=1,nsim
              feasible(j,jj)=.true.
225         continue
220       continue

              if (constflag .eqv. .true.) then
!c   PROJECT CATCH USING A CONSTANT F
!c   USE FTARGET, IF APPLICABLE
               if ((ftarflag .eqv. .true.) .and.(use_ftarget .eqv. .true.)) then
                 call calc_catch(n,nage,maxage,ftarget,pr,m,catch)
                 simf(boot,sim,time)=ftarget
               else
                 call calc_catch(n,nage,maxage,f,pr,m,catch)
                 simf(boot,sim,time)=f
               endif

              else
!c   PROJECT CATCH USING A TIME-VARYING F
!c   USE FTARGET, IF APPLICABLE
```

```
            if ((ftarflag .eqv. .true.) .and.(use_ftarget .eqv. .true.)) then
              call calc_catch(n,nage,maxage,ftarget,pr,m,catch)
              simf(boot,sim,time)=ftarget
            else
              call calc_catch(n,nage,maxage,fseries(time),pr,m,catch)
              simf(boot,sim,time)=fseries(time)
            endif
          endif
        endif

      else if (mixflag .eqv. .true.) then
!c   MIXTURE OF F AND QUOTA-BASED CATCHES
!c   DETERMINE WHETHER AN F OR A QUOTA IS APPLIED
        if (mixyr(time) .eq. 1) then

!c   QUOTA-BASED CATCH
!c   COMPUTE AN UPPER BOUND ON LANDINGS (SET F=fmax)
          call calc_catch(n,nage,maxage,fmax,pr,m,maxcatch)
          ubland=0.
!c   INCLUDE DISCARDS, IF APPLICABLE
          if (discflag .eqv. .true.) then
            do 395 j=1,nage
              ubland=ubland+maxcatch(j)*(1.-discfrac(j))*wtland(j)
395           continue
          else
            do 390 j=1,nage
              ubland=ubland+maxcatch(j)*wtland(j)
390           continue
          endif
!c   CHECK IF QUOTA IS FEASIBLE
          if (qseries(time) .gt. ubland) then
            feasible(boot,sim)=.false.
          else
            feasible(boot,sim)=.true.
          endif
!c   IF INFEASIBLE, EXIT THIS SIMULATION
          if (feasible(boot,sim) .eqv. .false.) goto 999
!c   COMPUTE F TO REALIZE THE QUOTA
          fquota=rtsafe(fmin,fmax,ftol)
!c   STORE THE REALIZED F
          simf(boot,sim,time)=fquota
!c   APPLY THIS F TO CALCULATE CATCH
          call calc_catch(n,nage,maxage,fquota,pr,m,catch)
```

```
          else if (mixyr(time) .eq. 0) then

!c   F-BASED CATCH
!c   PROJECT CATCH USING A TIME-VARYING F
!c   USE FTARGET, IF APPLICABLE
             if ((ftarflag .eqv. .true.) .and.(use_ftarget .eqv. .true.)) then
               call calc_catch(n,nage,maxage,ftarget,pr,m,catch)
               simf(boot,sim,time)=ftarget
             else
               call calc_catch(n,nage,maxage,fseries(time),pr,m,catch)
               simf(boot,sim,time)=fseries(time)
             endif
           endif
         endif
!c   END OF CATCH COMPUTATION
!c   COMPUTE LANDINGS-NOTE THAT simf STORES THE REALIZED F
         landings=0.0
         if (discflag .eqv. .true.) then
           do 315 j=1,nage
             landings=landings+catch(j)*wtland(j)*(1.-discfrac(j))
315          continue
         else
           do 320 j=1,nage
             landings=landings+catch(j)*wtland(j)
320          continue
         endif

!c   STORE THE LANDINGS
         simland(boot,sim,time)=landings

!c   COMPUTE MARKET CATEGORY SUMMARIES, IF APPLICABLE
         if (mcflag .eqv. .true.) then
             if(boot==1.and.sim==1.and.time==1) then
                 call mcAlloc
             endif

!c   CHECK IF DISCARDING IS INCLUDED IN PROJECTIONS
           if (discflag .eqv. .true.) then
!c   INCLUDE DISCARD FRACTION AT AGE

!c   SUM CATCH FOR MARKET CATEGORY 1
             mc_wt=0.0
             mc_num=0.0
             do 700 j=1,nage
```

```
            mc_wt=mc_wt+catch(j)*wtland(j)*market(1,j)*(1.0-discfrac(j))
700          continue
            do 705 j=1,nage
            mc_num=mc_num+catch(j)*market(1,j)*(1.0-discfrac(j))
705          continue
            simmc1_w(boot,sim,time)=mc_wt
            simmc1_n(boot,sim,time)=mc_num

!c   SUM CATCH FOR MARKET CATEGORY 2, IF APPLICABLE
            if (nmc .ge. 2) then
             mc_wt=0.0
             mc_num=0.0
             do 710 j=1,nage
              mc_wt=mc_wt+catch(j)*wtland(j)*market(2,j)*(1.0-discfrac(j))
710           continue
             do 715 j=1,nage
              mc_num=mc_num+catch(j)*market(2,j)*(1.0-discfrac(j))
715           continue
             simmc2_w(boot,sim,time)=mc_wt
             simmc2_n(boot,sim,time)=mc_num
            endif

!c   SUM CATCH FOR MARKET CATEGORY 3, IF APPLICABLE
            if (nmc .eq. 3) then
             mc_wt=0.0
             mc_num=0.0
             do 720 j=1,nage
              mc_wt=mc_wt+catch(j)*wtland(j)*market(3,j)*(1.0-discfrac(j))
720           continue
             do 725 j=1,nage
              mc_num=mc_num+catch(j)*market(3,j)*(1.0-discfrac(j))
725           continue
             simmc3_w(boot,sim,time)=mc_wt
             simmc3_n(boot,sim,time)=mc_num
            endif

           else
!c  THERE IS NO DISCARD FRACTION AT AGE

!c   SUM CATCH FOR MARKET CATEGORY 1
            mc_wt=0.0
            mc_num=0.0
            do 730 j=1,nage
             mc_wt=mc_wt+catch(j)*wtland(j)*market(1,j)
```

```
730           continue
             do 735 j=1,nage
              mc_num=mc_num+catch(j)*market(1,j)
735           continue
             simmc1_w(boot,sim,time)=mc_wt
             simmc1_n(boot,sim,time)=mc_num

!c   SUM CATCH FOR MARKET CATEGORY 2, IF APPLICABLE
             if (nmc .ge. 2) then
              mc_wt=0.0
              mc_num=0.0
              do 740 j=1,nage
               mc_wt=mc_wt+catch(j)*wtland(j)*market(2,j)
740            continue
              do 745 j=1,nage
               mc_num=mc_num+catch(j)*market(2,j)
745            continue
              simmc2_w(boot,sim,time)=mc_wt
              simmc2_n(boot,sim,time)=mc_num
             endif

!c   SUM CATCH FOR MARKET CATEGORY 3, IF APPLICABLE
             if (nmc .eq. 3) then
              mc_wt=0.0
              mc_num=0.0
              do 750 j=1,nage
               mc_wt=mc_wt+catch(j)*wtland(j)*market(3,j)
750            continue
              do 755 j=1,nage
               mc_num=mc_num+catch(j)*market(3,j)
755            continue
              simmc3_w(boot,sim,time)=mc_wt
              simmc3_n(boot,sim,time)=mc_num
             endif
            endif
           endif
!          endif

!c   COMPUTE DISCARDS, IF APPLICABLE
           if (discflag .eqv. .true.) then
            discards=0.0
            do 325 j=1,nage
             discards=discards+catch(j)*discfrac(j)*wtdisc(j)
325           continue
```

```
!c  STORE THE DISCARDS, IF APPLICABLE
          simdisc(boot,sim,time)=discards
        endif

!c  AGE-2 RECRUITMENT
!c  FOR THE INITIAL YEAR, COMPUTE SSB IN PREVIOUS YEAR
!c  NOTE THAT past_ssb GENERATES RECRUITMENT
!c  ALSO NOTE THAT IF pr, m, OR zproj ARE CONSTANT, THEN
!c  past_pr=pr, past_m=m, AND past_zproj=zproj
        if ((age2recflag .eqv. .true.) .and. (time .eq. 1)) then

past_ssb=calc_ssb(past_n,nage,maxage,past_pr,past_m,boot_f(boot),past_zproj,fm,wt)
        endif

!c      REVISED 7/1/99
!c  COMPUTE MEAN B
        meanB=calc_meanB(n,lowerage,upperage,maxage,pr,m,simf(boot,sim,time),wtland)

!c  STORE MEAN B
        simmeanB(boot,sim,time)=meanB

!c      REVISED FEB-2002
!c  COMPUTE TOTAL B
        totB=calc_totB(n,maxage,wt)

!c  STORE TOTAL B
        simtotB(boot,sim,time)=totB

!c      REVISED 7/8/99
!c  COMPUTE FmeanB
        FmeanB=calc_FmeanB(n,lowerage,upperage,maxage,pr,m,simf(boot,sim,time),wtland)

!c  STORE FmeanB
        simFmeanB(boot,sim,time)=FmeanB

!c  COMPUTE SSB
        ssb=calc_ssb(n,nage,maxage,pr,m,simf(boot,sim,time),zproj,fm,wt)

!c  STORE SSB
        simssb(boot,sim,time)=ssb

!c  SET use_ftarget FLAG BASED ON ssb, IF APPLICABLE
        year=baseyr+time-1
```

```
      if ((ftarflag .eqv. .true.) .and.(year .ge. ftaryear) .and.(ssb .ge. ssbthresh)) then
       use_ftarget=.true.
      else
       use_ftarget=.false.
      endif

!c   SET ssb_value TO ssb OR past_ssb BASED ON RECRUITMENT AGE
      if (age2recflag .eqv. .false.) then
       ssb_value=ssb
      else
       ssb_value=past_ssb
      endif

!c   GENERATE RECRUITMENT
      if (recflag .eq. 1) then
!c   USE MODEL 1
!c   DETERMINE SSB STATE (ssbindex)
       done=.false.
       index=0
       do while (done .eqv. .false.)
        index=index+1
        if (index .eq. nssb) then
         done=.true.
         ssbindex=index
        else if (ssb_value .lt. ssb_cut(index)) then
         done=.true.
         ssbindex=index
        endif
       end do

!c   GENERATE STOCHASTIC RECRUITMENT (reclevel)
       uniform=ran2(idum)
       sum=0.0
       index=0
       done=.false.
       do while (done .eqv. .false.)
        index=index+1
        sum=sum+p_reclevel(ssbindex,index)
        if (uniform .lt. sum) then
         done=.true.
         reclevel=recruit(index)
        endif
        if (index .gt. maxrec) then
```

```
              if (abs(uniform-sum) .lt. 0.01) then
                done=.true.
                reclevel=recruit(nrec)
              else
                print *,'ERROR GENERATING RECRUITMENT'
                done=.true.
                reclevel=recruit(nrec)
              endif
            endif
          end do

        else if (recflag .eq. 2) then
!c   USE MODEL 2
          uniform=ran2(idum)
!c   DETERMINE INDEX FOR INTERPOLATION
          index=int(uniform*(nrec-1))+1
          ssbindex=maxobsrec-nrec+index
!c   GENERATE STOCHASTIC RECRUITMENT (reclevel)
          sum=uniform-(real(index-1)/real(nrec-1))
!c   EQUATION 19

reclevel=sum*real(nrec-1)*(sortrssb(ssbindex+1)-sortrssb(ssbindex))+sortrssb(ssbindex)
!c   EQUATION 24
          reclevel=reclevel*ssb_value

!c       REVISED FEB-2002
        else if (recflag .eq. 14) then
!c   USE MODEL 14
          uniform=ran2(idum)
!c   DETERMINE INDEX FOR INTERPOLATION
          index=int(uniform*(nrec-1))+1
          ssbindex=maxobsrec-nrec+index
!c   GENERATE STOCHASTIC RECRUITMENT (reclevel)
          sum=uniform-(real(index-1)/real(nrec-1))
!c   EQUATION 19
          reclevel=sum*real(nrec-1)*(sortrec(ssbindex+1)-sortrec(ssbindex))+sortrec(ssbindex)

        else if (recflag .eq. 3) then
!c   USE MODEL 3
          uniform=ran2(idum)
          ssbindex=int(uniform*nrec)+1
          reclevel=obsrec(ssbindex)
        else if (recflag .eq. 9) then
!C   USE MODEL 9
```

```
          uniform=ran2(idum)
          ssbindex=int(uniform*nrec)+1
          reclevel=obsrec3(time,ssbindex)

        else if (recflag .eq. 4) then
!c   USE MODEL 4
          uniform=ran2(idum)
!c   DETERMINE SSB STATE (LOW OR HIGH)
          if (ssb_value .lt. ssbcut_m4) then
!c   DETERMINE INDEX FOR INTERPOLATION WITH LOW SSB
            index=int(uniform*(nrec1-1))+1
            ssbindex=maxobsrec-nrec1+index
!c   GENERATE STOCHASTIC RECRUITMENT (reclevel)
            sum=uniform-(real(index-1)/real(nrec1-1))
!c   EQUATION 19
            reclevel=sum*real(nrec1-1)*(sort1(ssbindex+1)-sort1(ssbindex))+sort1(ssbindex)
          else
!c   DETERMINE INDEX FOR INTERPOLATION WITH HIGH SSB
            index=int(uniform*(nrec2-1))+1
            ssbindex=maxobsrec-nrec2+index
!c   GENERATE STOCHASTIC RECRUITMENT (reclevel)
            sum=uniform-(real(index-1)/real(nrec2-1))
!c   EQUATION 19
            reclevel=sum*real(nrec2-1)*(sort2(ssbindex+1)-sort2(ssbindex))+sort2(ssbindex)
          endif
!c   EQUATION 24
          reclevel=reclevel*ssb_value

!c       REVISED FEB-2002
        else if (recflag .eq. 15) then
!c   USE MODEL 15
          uniform=ran2(idum)
!c   DETERMINE SSB STATE (LOW OR HIGH)
          if (ssb_value .lt. ssbcut_m4) then
!c   DETERMINE INDEX FOR INTERPOLATION WITH LOW SSB
            index=int(uniform*(nrec1-1))+1
            ssbindex=maxobsrec-nrec1+index
!c   GENERATE STOCHASTIC RECRUITMENT (reclevel)
            sum=uniform-(real(index-1)/real(nrec1-1))
!c   EQUATION 19
            reclevel=sum*real(nrec1-1)*(sort1(ssbindex+1)-sort1(ssbindex))+sort1(ssbindex)
          else
!c   DETERMINE INDEX FOR INTERPOLATION WITH HIGH SSB
            index=int(uniform*(nrec2-1))+1
```

```
            ssbindex=maxobsrec-nrec2+index
!c   GENERATE STOCHASTIC RECRUITMENT (reclevel)
            sum=uniform-(real(index-1)/real(nrec2-1))
!c   EQUATION 19
            reclevel=sum*real(nrec2-1)*(sort2(ssbindex+1)-sort2(ssbindex))+sort2(ssbindex)
           endif

         else if (((recflag .ge. 5) .and. (recflag .le. 8)) .or. ((recflag .ge. 10) .and. (recflag .le. 13)))
then
!c   LOGNORMAL MODELS 5, 6, 7, AND 8, OR 10, 11, 12, AND 13
!c   EXPRESS SSB (kg) IN APPROPRIATE UNITS FOR SRR
           ssb_srr=ssb_value/ssb_unit

           if ((recflag .eq. 5) .or. (recflag .eq. 10)) then
!c   BEVERTON-HOLT SRR
            sum=a_bhsrr*ssb_srr/(b_bhsrr+ssb_srr)

           else if ((recflag .eq. 6) .or. (recflag .eq. 11)) then
!c   RICKER SRR
            sum=a_ricsrr*ssb_srr*exp(-b_ricsrr*ssb_srr)

           else if ((recflag .eq. 7) .or. (recflag .eq. 12)) then
!c   SHEPHERD SRR
            sum=1.0+((ssb_srr/k_shsrr)**b_shsrr)
            sum=a_shsrr*ssb_srr/sum

           else if ((recflag .eq. 8) .or. (recflag .eq. 13)) then
!c   LOGNORMAL DISTRIBUTION
            sum=1.0
           endif

!c   DO LOOP TO ENSURE CONSTRAINED R/SSB VALUES
           done=.false.

           do while (done .eqv. .false.)

!c   GENERATE UNIT NORMAL RANDOM VARIATE
            z=gasdev(idum)

!c   IF UNCORRELATED (MODELS 5-7), TRANSFORM z WITH RESIDUAL VARIANCE
            if ((recflag .ge. 5) .and. (recflag .le. 7)) then

!c   TRANSFORM UNIT NORMAL VARIATE TO N(0,residvar) VARIATE
             z=z*(residvar)**0.5
```

```
!c  ELSE IF CORRELATED (MODELS 5-7), TRANSFORM z WITH SIGMAW VARIANCE
        else if ((recflag .ge. 10) .and. (recflag .le. 12)) then

!c  TRANSFORM UNIT NORMAL VARIATE TO N(0,sigmasqw) VARIATE
          z=z*(sigmasqw)**0.5

        else if ((recflag .eq. 8) .or. (recflag .eq. 13)) then

!c  ELSE TRANSFORM z TO N(lnrec_mean,lnrec_std^2) VARIATE
          z=lnrec_mean+z*lnrec_std

        endif

!c  EXPONENTIATE IF UNCORRELATED
        if ((recflag .ge. 5) .and. (recflag .le. 8)) then
          z=exp(z)

!c  ELSE INCLUDE PREVIOUS RESIDUAL
        else if ((recflag .ge. 10) .and. (recflag .le. 13)) then

            z=z+phi*resid(time-1)

            z=exp(z)

        endif

!c  APPLY LOGNORMAL VARIATE TO sum
        reclevel=sum*z

!c  COMPUTE RESIDUAL
        if ((recflag .ge. 10) .and. (recflag .le. 13)) then

            resid(time)=log(reclevel)-log(sum)

          endif

!c  EXPRESS RECRUITMENT (rec_unit) IN ABSOLUTE NUMBERS OF FISH
        reclevel=reclevel*rec_unit

!c  CALCULATE REALIZED R/SSB VALUE
        z=reclevel/ssb_value

!c  APPLY R/SSB CONSTRAINTS, IF APPLICABLE
```

```
            if (bdrecflag .eqv. .true.) then
              if (ssb_value .lt. ssb_min) then
!c   IF SSB IS BELOW MINIMUM OBSERVED VALUE, USE BOUNDS
                if ((z .ge. rssb_lower) .and.(z .le. rssb_upper)) then
                  done=.true.
                endif
!c   ELSE USE MIN AND MAX OBSERVED R/SSB VALUES AS CUTOFFS
              else if (ssb_value .ge. ssb_min) then
                if ((z .ge. rssb_min) .and. (z .le. rssb_max)) then
                  done=.true.
                endif
              endif
            else
!c   DO NOT APPLY R/SSB CONSTRAINTS
              done=.true.
            endif
          end do
        endif

!c  STORE THE REALIZED RECRUITMENT
        simrecr(boot,sim,time)=reclevel/bootunit

        if (indexflag .eqv. .true.) then
!c  STORE THE REALIZED VALUE FOR AGE index_age AND CONVERT TO INDEX
          simsvind(boot,sim,time)=a_intercept +b_slope*n(index_age)/bootunit
        endif

!c   COMPUTE NEXT POPULATION VECTOR
        call calc_next_n(n,nage,pr,m,simf(boot,sim,time),reclevel,next_n)

!c   COPY ssb TO past_ssb
        past_ssb=ssb

!c   COPY NEXT POPULATION VECTOR TO CURRENT POPULATION VECTOR
        do 330 j=1,nage
          n(j)=next_n(j)
330       continue

300     continue
!c   END OF TIME HORIZON LOOP

!c   JUMP POINT TO EXIT TIME LOOP WHEN QUOTA LEVEL IS INFEASIBLE
999     continue
```

```
200     continue
!c   END OF SIMULATION LOOP

100   continue
!c   END OF LOOP OVER BOOTSTRAP REPLICATES

!c        REVISED 7/8/99 NO EXTERNAL FILES CREATED WITHOUT A FLAG
!c        open(unit=89,file=SSBoutfile,status='unknown')
!c        open(unit=92,file=meanBoutfile,status='unknown')
!c        open(unit=90,file=Foutfile,status='unknown')
!c        open(unit=91,file=Landoutfile,status='unknown')

!c        do 1003 sim=1, nsim
!c                do 1002 boot=1,nboot
!c                        write(89,*)(simssb(boot,sim,time)/1000,time=1,ntime)
!c                        write(92,*)(simmeanB(boot,sim,time)/1000,time=1,ntime)
!c                        write(90,*)(simf(boot,sim,time),time=1,ntime)
!c                        write(91,*)(simland(boot,sim,time),time=1,ntime)
!c                1002 continue
!c        1003 continue
!c        close(89)
!c        close(92)
!c        close(90)
!c        close(91)

!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!c          SUMMARIZE RESULTS
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

!c   COMPUTE NUMBER OF FEASIBLE SIMULATIONS FOR AVERAGING
     allfeasible = .true.
     if ((quotaflag .eqv. .true.) .or. (mixflag .eqv. .true.)) then
       nfeasible=0
       do 400 j=1,nboot
         do 410 jj=1,nsim
           if (feasible(j,jj) .eqv. .true.) then
             nfeasible=nfeasible+1
           endif
410        continue
400     continue

!c        REVISED JUL-2002
        if (nfeasible .lt. (nsim*nboot)) then
            allfeasible = .false.
```

```
          endif
        p_feasible=real(nfeasible)/real((nsim*nboot))
      else
!c   IF PROJECTIONS ARE F_BASED, ALL REALIZATIONS ARE FEASIBLE
        nfeasible=nsim*nboot
          p_feasible=1.000000
      endif

!c   EXIT IF NO FEASIBLE SIMULATIONS
      if (nfeasible .eq. 0) then
        print *,' '
        print *,'All simulations are infeasible: Landings quota is too large.'
        print *,' '
        print *,'Exiting projection analysis.'
        print *,' '
        goto 99999
      endif

!c  SET THE OFFSET INDEX RELATED TO THE NUMBER OF INFEASIBLE
SOLUTIONS
      iflag=maxlen-nfeasible

!c   REVISED 7/8/99
!c        REVISED FEB-2002
!c  COUNT THE NUMBER OF TIMES THAT SSB and MEAN B and F_WT_B
!c  EXCEED THRESHOLD VALUES (ssbthresh & meanBthresh & FmeanBthresh)
!c  INCLUDE Fthresh and totBthresh
      if (sfaflag .eqv. .true.) then
        do 440 j=1,nboot
          do 445 jj=1,nsim
            if ((feasible(j,jj)) .eqv. .true.) then
              do 450 jjj=1,ntime
                ssb = simssb(j,jj,jjj)
                if (ssb .ge. ssbthresh) then
                  ssb_count(jjj)=ssb_count(jjj)+1.0
                endif
                meanB = simmeanB(j,jj,jjj)
                if (meanB .ge. meanBthresh) then
                  meanB_count(jjj)=meanB_count(jjj)+1.0
                endif
                totB = simtotB(j,jj,jjj)
                if (totB .ge. totBthresh) then
                  totB_count(jjj)=totB_count(jjj)+1.0
                endif
```

```
              FmeanB = simFmeanB(j,jj,jjj)
              if (FmeanB .ge. FmeanBthresh) then
                FmeanB_count(jjj)=FmeanB_count(jjj)+1.0
              endif
              tmpF = simf(j,jj,jjj)
              if (tmpF .ge. Fthresh) then
                F_count(jjj)=F_count(jjj)+1.0
              endif
450           continue
            endif
445       continue
440     continue

!c   COMPUTE CONDITIONAL PROBABILITY THAT SSB and MEAN B and F_WT_B
!c   EXCEED THRESHOLD VALUES IN EACH YEAR (CONDITIONED ON FEASIBLE
TRAJECTORIES)
!c   INCLUDE F and totB
       do 455 j=1, ntime
         p_ssbthresh(j)=ssb_count(j) / real(nfeasible)
         p_meanBthresh(j)=meanB_count(j) / real(nfeasible)
         p_totBthresh(j)=totB_count(j) / real(nfeasible)
         p_FmeanBthresh(j)=FmeanB_count(j) / real(nfeasible)
         p_Fthresh(j)=F_count(j) / real(nfeasible)
455     continue

!c        REVISED JUL-2002
!c   COMPUTE PROBABILITY THAT SSB, MEAN_B, F_WT_B, F, and TOT_B
!c   EXCEED THRESHOLD VALUES IN EACH YEAR (USE ALL TRAJECTORIES)
       if (allfeasible .eqv. .false.) then
         do 456 j=1, ntime
           pjoint_ssbthresh(j)=p_ssbthresh(j)*p_feasible
           pjoint_meanBthresh(j)=p_meanBthresh(j)*p_feasible
              pjoint_totBthresh(j)=p_totBthresh(j)*p_feasible
           pjoint_FmeanBthresh(j)=p_FmeanBthresh(j)*p_feasible+(1.0-p_feasible)
           pjoint_Fthresh(j)=p_Fthresh(j)*p_feasible+(1.0-p_feasible)
456       continue
       endif
     endif


!c  COUNT THE NUMBER OF TIMES THAT THE RECRUITMENT INDEX
!c  EXCEEDS THE CRITICAL VALUE
     if (indexflag .eqv. .true.) then
       do 420 j=1,nboot
```

```
        do 425 jj=1,nsim
          if ((feasible(j,jj)).eqv. .true.) then
            do 430 jjj=1,ntime
              res_n2 = simsvind(j,jj,jjj)
              if (res_n2 .ge. crit_index) then
                crit_count(jjj)=crit_count(jjj)+1.0
              endif
430           continue
            endif
425       continue
420     continue

!c   COMPUTE CONDITIONAL PROBABILITY THAT RECRUITMENT INDEX
!c   EXCEEDS CRITICAL VALUE IN EACH YEAR
       do 435 j=1, ntime
         p_index(j)=crit_count(j) / real(nfeasible)
435     continue

!c        REVISED JUL-2002
!c   COMPUTE PROBABILITY THAT RECRUITMENT INDEX
!c   EXCEEDS CRITICAL VALUE IN EACH YEAR
       if (allfeasible .eqv. .false.) then
         do 436 j=1, ntime
           pjoint_index(j)=p_index(j)*p_feasible
436       continue
       endif
     endif

!c        REVISED 7/1/99
!c        REVISED FEB-2002
!c SUMMARIZE THE FOLLOWING SIMULATED OUTPUTS:
!c
!c         OUTPUT                          VARIABLE NAME
!c
!c     1.  SPAWNING STOCK BIOMASS (MT)                 simssb
!c     2.  MEAN STOCK BIOMASS (MT)               simmeanB
!c         OF AGES lowerage to upperage
!c     3a.  F WEIGHTED BY MEAN BIOMASS           simFmeanB
!c         OF AGES lowerage to upperage
!c     3b.  TOTAL STOCK BIOMASS (MT)             simtotB
!c     4.  RECRUITMENT     (#)                 simrecr
!c     5.  SURVEY INDEX FOR AGE index_age      simsvind
!c     6.  DISCARDS (MT)                           simdisc
!c     7.  LANDINGS (MT) {F-based only}              simland
```

```
!c    8.  REALIZED F {quota-based only}              simf
!c    9.  MARKET CATEGORY SUMMARIES                  simmc*_w,simmc*_n

!c       REVISED 7/1/99
      call summarize(simmeanB,avgmeanB,sdmeanB,pctmeanB,quotaflag,mixflag)

!c       REVISED 7/8/99
      call summarize(simFmeanB,avgFmeanB,sdFmeanB,pctFmeanB,quotaflag,mixflag)

!c       REVISED FEB-2002
      call summarize(simtotB,avgtotB,sdtotB,pcttotB,quotaflag,mixflag)

      call summarize(simssb,avgssb,sdssb,pctssb,quotaflag,mixflag)
      call summarize(simrecr,avgrecr,sdrecr,pctrecr,quotaflag,mixflag)
      call summarize(simsvind,avgsvind,sdsvind,pctsvind,quotaflag,mixflag)
      call summarize(simdisc,avgdisc,sddisc,pctdisc,quotaflag,mixflag)

!c   COMPUTE TIME-AVERAGES OF LANDINGS FOR F-BASED PROJECTIONS
      if ((quotaflag .eqv. .false.) .and. (mixflag .eqv. .false.)) then
        call summarize(simland,avgland,sdland,pctland,quotaflag,mixflag)
      endif

!c   COMPUTE TIME-AVERAGES OF REALIZED F'S FOR QUOTA-BASED
PROJECTIONS
      if ((quotaflag .eqv. .true.) .and. (mixflag .eqv. .false.)) then
        call summarize(simf,avgf,sdf,pctf,quotaflag,mixflag)
      endif

!c   COMPUTE TIME-AVERAGES OF LANDINGS AND QUOTAS
      if ((mixflag .eqv. .true.) .or. (ftarflag .eqv. .true.)) then
        call summarize(simland,avgland,sdland,pctland,quotaflag,mixflag)
        call summarize(simf,avgf,sdf,pctf,quotaflag,mixflag)
      endif

!c   COMPUTE TIME-AVERAGES OF LANDINGS BY MARKET CATEGORY
      if (mcflag .eqv. .true.) then
        call summarize(simmc1_w,avgmc1_w,sdmc1_w,pctmc1_w,quotaflag,mixflag)
        call summarize(simmc1_n,avgmc1_n,sdmc1_n,pctmc1_n,quotaflag,mixflag)
        if (nmc .ge. 2) then
          call summarize(simmc2_w,avgmc2_w,sdmc2_w,pctmc2_w,quotaflag,mixflag)
          call summarize(simmc2_n,avgmc2_n,sdmc2_n,pctmc2_n,quotaflag,mixflag)
        endif
        if (nmc .eq. 3) then
          call summarize(simmc3_w,avgmc3_w,sdmc3_w,pctmc3_w,quotaflag, mixflag)
```

```
        call summarize(simmc3_n,avgmc3_n,sdmc3_n,pctmc3_n,quotaflag,mixflag)
      endif

!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!c          OUTPUT RESULTS
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

!c   WRITE SUMMARIES TO OUTPUT FILE
!c   OUTPUT AVERAGES AND MEDIANS BY YEAR
       open(unit=2,file=mcfile,status='unknown')
!c   1, 2, or 3 MARKET CATEGORIES
       if (nmc .eq. 1) then
         write(2,*) 'AVERAGE TOTAL WEIGHT (KG) AND NUMBERS BY YEAR'
         do 800 j=1,ntime
           write(2,'(i4,2(1x,f10.0))') (baseyr+j-1),avgmc1_w(j),avgmc1_n(j)
800       continue
         write(2,*) 'MEDIAN TOTAL WEIGHT (KG) AND NUMBERS BY YEAR'
         do 802 j=1,ntime
           write(2,'(i4,2(1x,f10.0))') (baseyr+j-1),pctmc1_w(j,5),pctmc1_n(j,5)
802       continue

       else if (nmc .eq. 2) then
         write(2,*) 'AVERAGE TOTAL WEIGHT (KG) AND NUMBERS BY YEAR'
         do 810 j=1,ntime
           write(2,'(i4,4(1x,f10.0))')
(baseyr+j-1),avgmc1_w(j),avgmc1_n(j),avgmc2_w(j),avgmc2_n(j)
810       continue
         write(2,*) 'MEDIAN TOTAL WEIGHT (KG) AND NUMBERS BY YEAR'
         do 812 j=1,ntime
           write(2,'(i4,4(1x,f10.0))')
(baseyr+j-1),pctmc1_w(j,5),pctmc1_n(j,5),pctmc2_w(j,5),pctmc2_n(j,5)
812       continue

       else if (nmc .eq. 3) then
         write(2,*) 'AVERAGE TOTAL WEIGHT (KG) AND NUMBERS BY YEAR'
         do 820 j=1,ntime
           write(2,'(i4,6(1x,f10.0))')
(baseyr+j-1),avgmc1_w(j),avgmc1_n(j),avgmc2_w(j),avgmc2_n(j),avgmc3_w(j),avgmc3_n(j)
820       continue
         write(2,*) 'MEDIAN TOTAL WEIGHT (KG) AND NUMBERS BY YEAR'
         do 822 j=1,ntime
         write(2,'(i4,6(1x,f10.0))')
baseyr+j-1,pctmc1_w(j,5),pctmc1_n(j,5),pctmc2_w(j,5),pctmc2_n(j,5),pctmc3_w(j,5),pctmc3_n
(j,5)
```

```
822       continue
        endif

        write(2,*) 'SIMULATED AGGREGATE LANDINGS BY MARKET CATEGORY'
        if (nmc .eq. 1) then
          do 900 time=1,ntime
            write(2,*) 'TIME=',(baseyr+time-1)
            do 905 boot=1,nboot
              do 910 sim=1,nsim
                write(2,'(2(1x,f10.0))') simland(boot,sim,time),simmc1_w(boot,sim,time)
910           continue
905         continue
900       continue

        else if (nmc .eq. 2) then
          do 920 time=1,ntime
            write(2,*) 'TIME=',(baseyr+time-1)
            do 925 boot=1,nboot
              do 930 sim=1,nsim
                write(2,'(3(1x,f10.0))')
simland(boot,sim,time),simmc1_w(boot,sim,time),simmc2_w(boot,sim,time)
930           continue
925         continue
920       continue

        else if (nmc .eq. 3) then
          do 940 time=1,ntime
            write(2,*) 'TIME=',(baseyr+time-1)
            do 945 boot=1,nboot
              do 950 sim=1,nsim
             write(2,'(4(1x,f10.0))')
simland(boot,sim,time),simmc1_w(boot,sim,time),simmc2_w(boot,sim,time),simmc3_w(boot,si
m,time)
950           continue
945         continue
940       continue
        endif
        close(2)
      endif

!c    WRITE RESULTS TO OUTPUT FILE
      open(unit=2,file=outfile,status='unknown')
      write(2,*) 'PROJECTION RUN: ',runname
      write(2,*) 'INPUT FILE: ',infile
```

```
      write(2,*) 'OUTPUT FILE: ',outfile
      write(2,*) 'RECRUITMENT MODEL: ',recflag
      write(2,*) 'NUMBER OF BOOTSTRAP REALIZATIONS: ',nboot
      write(2,*) 'NUMBER OF SIMULATIONS PER BOOTSTRAP REALIZATION: ',nsim
      write(2,*) 'TOTAL NUMBER OF SIMULATIONS: ',(nsim*nboot)
      write(2,*) 'NUMBER OF FEASIBLE SIMULATIONS: ',nfeasible
      write(2,*) 'PROPORTION OF SIMULATIONS THAT ARE FEASIBLE: ',p_feasible
      write (2,*) ' '

      if (mixflag .eqv. .false.) then
       if (quotaflag .eqv. .true.) then
        write(2,*) 'QUOTA-BASED PROJECTIONS'
        if (constflag .eqv. .true.) then
         write(2,'(a,f10.3)') 'CONSTANT QUOTA (000 MT):',(quota/1.0e6)
        else
         write(2,*) 'TIME-VARYING QUOTA'
         write(2,*) 'YEAR    QUOTA (000 MT)'
         do 500 j=1,ntime
          write(2,'(1x,i4,1x,f10.3)') (baseyr+j-1),(qseries(j)/1.0e6)
500       continue
        endif
       else
        write(2,*) 'F-BASED PROJECTIONS'
        if (constflag .eqv. .true.) then
         write(2,'(a,f5.3)') 'CONSTANT F:',f
        else
         if (ftarflag .eqv. .true.) then
          write(2,*) 'F-SERIES IF SSB THRESHOLD IS NEVER EXCEEDED'
         endif
         write(2,*) 'TIME-VARYING F'
         write(2,*) 'YEAR    F'
         do 510 j=1,ntime
          write(2,'(1x,i4,4x,f5.3)') (baseyr+j-1),fseries(j)
510       continue
        endif
       endif

      else
!c   MIXTURE OF F AND QUOTA-BASED CATCHES
       write(2,*) 'MIXTURE OF F AND QUOTA BASED CATCHES'
       write(2,*) 'YEAR   F       QUOTA (THOUSAND MT)'
       do 5300 j=1,ntime
        if (mixyr(j) .eq. 1) then
          write(2,'(1x,i4,a,f10.3)') (baseyr+j-1),'       ',(qseries(j)/1.0e6)
```

```
        else
           write(2,'(1x,i4,2x,f5.3)') (baseyr+j-1),fseries(j)
        endif
5300    continue
     endif

     write(2,*) ' '
     write(2,*) 'SPAWNING STOCK BIOMASS (THOUSAND MT)'
     write(2,*) 'YEAR    AVG SSB (000 MT)    STD'
     do 520 j=1,ntime
       write(2,'(1x,i4,4x,f10.3,4x,f10.3)') (baseyr+j-1),(avgssb(j)/1.0e6),(sdssb(j)/1.0e6)
520   continue
     write(2,*) ' '
     write(2,*) 'PERCENTILES OF SPAWNING STOCK BIOMASS (000 MT)'
     write(2,*) 'YEAR       1%        5%        10%       25%       50%       75%       90%       95%
      99%'
     do 525 j=1,ntime
       write(2,'(1x,i4,9(1x,f10.3))') (baseyr+j-1),((pctssb(j,k)/1.0e6),k=1,maxpct)
525   continue
     write(2,*) ' '

     if (sfaflag .eqv. .true.) then
       write(2,'(a,f10.3,a)') 'ANNUAL PROBABILITY THAT SSB EXCEEDS
THRESHOLD:',(ssbthresh/1.0e6),' THOUSAND MT'
       write(2,*) 'YEAR    Pr(SSB > Threshold Value) FOR FEASIBLE SIMULATIONS'
       do 528 j=1,ntime
         write(2,'(1x,i4,12x,f5.3)') (baseyr+j-1),p_ssbthresh(j)
528     continue
       write(2,*) ' '
!c         REVISED JUL-2002
          if (allfeasible .eqv. .false.) then
         write(2,'(a,f10.3,a)') 'JOINT PROBABILITY THAT SSB EXCEEDS
THRESHOLD:',(ssbthresh/1.0e6),' THOUSAND MT'
         write(2,*) 'YEAR    Pr(SSB > Threshold Value)*Pr(Feasible)'
         do 529 j=1,ntime
           write(2,'(1x,i4,12x,f5.3)') (baseyr+j-1),pjoint_ssbthresh(j)
529       continue
       write(2,*) ' '
          endif
     endif

!c         REVISED 7/1/99
     write(2,*) ' '
     write(2,*) 'MEAN BIOMASS (THOUSAND MT) FOR AGES:',lowerage,' TO ',upperage
```

```
      write(2,*) 'YEAR   AVG MEAN B (000 MT)      STD'
      do 52044 j=1,ntime
       write(2,'(1x,i4,4x,f10.3,8x,f10.3)') (baseyr+j-1),(avgmeanB(j)/1.0e6),(sdmeanB(j)/1.0e6)
52044 continue
      write(2,*) ' '
      write(2,*) 'PERCENTILES OF MEAN STOCK BIOMASS (000 MT)'
      write(2,*) 'YEAR       1%        5%        10%       25%       50%       75%       90%       95%
       99%'
      do 52545 j=1,ntime
       write(2,'(1x,i4,9(1x,f10.3))') (baseyr+j-1),((pctmeanB(j,k)/1.0e6),k=1,maxpct)
52545 continue

!c        REVISED 7/8/99
      write(2,*) ' '
      if (sfaflag .eqv. .true.) then
       write(2,'(a,f10.3,a)') 'ANNUAL PROBABILITY THAT MEAN BIOMASS EXCEEDS
THRESHOLD:',(meanBthresh/1.0e6),' THOUSAND MT'
       write(2,*) 'YEAR    Pr(MEAN B > Threshold Value) FOR FEASIBLE SIMULATIONS'
       do 52899 j=1,ntime
        write(2,'(1x,i4,12x,f5.3)') (baseyr+j-1),p_meanBthresh(j)
52899   continue
       write(2,*) ' '
!c         REVISED JUL-2002
          if (allfeasible .eqv. .false.) then
         write(2,'(a,f10.3,a)') 'JOINT PROBABILITY THAT MEAN BIOMASS EXCEEDS
THRESHOLD:',(meanBthresh/1.0e6),' THOUSAND MT'
         write(2,*) 'YEAR    Pr(MEAN B > Threshold Value)*Pr(Feasible)'
         do 52897 j=1,ntime
          write(2,'(1x,i4,12x,f5.3)') (baseyr+j-1),pjoint_meanBthresh(j)
52897     continue
         write(2,*) ' '
          endif
      endif

!c        REVISED 7/8/99
      write(2,*) ' '
      write(2,*) 'F WEIGHTED BY MEAN BIOMASS FOR AGES:',lowerage,' TO ',upperage
      write(2,*) 'YEAR  AVG F_WT_B        STD'
      do 55099 j=1,ntime
       write(2,'(1x,i4,4x,f5.3,10x,f5.3)') (baseyr+j-1),avgFmeanB(j),sdFmeanB(j)
55099 continue
      write(2,*) ' '
      write(2,*) 'PERCENTILES OF F WEIGHTED BY MEAN BIOMASS FOR
AGES:',lowerage,' TO ',upperage
```

```
      write(2,*) 'YEAR   1%     5%     10%    25%    50%    75%    90%        95%     99%'
      do 55599 j=1,ntime
        write(2,'(1x,i4,9(1x,f6.3))') (baseyr+j-1),(pctFmeanB(j,k),k=1,maxpct)
55599 continue
      write(2,*) ' '
      if (sfaflag .eqv. .true.) then
        write(2,'(a,f6.3,a)') 'ANNUAL PROBABILITY THAT F WEIGHTED BY MEAN
BIOMASS EXCEEDS THRESHOLD:',FmeanBthresh
        write(2,*) 'YEAR    Pr(F_WT_B > Threshold Value) FOR FEASIBLE SIMULATIONS'
        do 52890 j=1,ntime
          write(2,'(1x,i4,12x,f5.3)') (baseyr+j-1),p_FmeanBthresh(j)
52890   continue
        write(2,*) ' '
!c        REVISED JUL-2002
          if (allfeasible .eqv. .false.) then
         write(2,'(a,f6.3,a)') 'JOINT PROBABILITY THAT F WEIGHTED BY MEAN
BIOMASS EXCEEDS THRESHOLD:',FmeanBthresh
         write(2,*) 'YEAR    Pr(F_WT_B > Threshold Value)*Pr(Feasible)+(1-Pr(Feasible))'
         do 52893 j=1,ntime
           write(2,'(1x,i4,12x,f5.3)') (baseyr+j-1),pjoint_FmeanBthresh(j)
52893     continue
         write(2,*) ' '
           endif
      endif

!c        REVISED FEB-2002
      write(2,*) ' '
      write(2,*) 'TOTAL STOCK BIOMASS (THOUSAND MT)'
      write(2,*) 'YEAR   AVG TOTAL B (000 MT)      STD'
      do 62044 j=1,ntime
        write(2,'(1x,i4,4x,f10.3,8x,f10.3)') (baseyr+j-1),(avgtotB(j)/1.0e6),(sdtotB(j)/1.0e6)
62044 continue
      write(2,*) ' '
      write(2,*) 'PERCENTILES OF TOTAL STOCK BIOMASS (000 MT)'
      write(2,*) 'YEAR       1%         5%         10%        25%        50%        75%        90%        95%
       99%'
      do 62545 j=1,ntime
        write(2,'(1x,i4,9(1x,f10.3))') (baseyr+j-1),((pcttotB(j,k)/1.0e6),k=1,maxpct)
62545 continue

      write(2,*) ' '
      if (sfaflag .eqv. .true.) then
        write(2,'(a,f10.3,a)') 'ANNUAL PROBABILITY THAT TOTAL STOCK BIOMASS
EXCEEDS THRESHOLD:',(totBthresh/1.0e6),' THOUSAND MT'
```

```
      write(2,*) 'YEAR    Pr(B > Threshold Value) FOR FEASIBLE SIMULATIONS'
      do 62899 j=1,ntime
       write(2,'(1x,i4,12x,f5.3)') (baseyr+j-1),p_totBthresh(j)
62899   continue
      write(2,*) ' '
!c       REVISED JUL-2002
         if (allfeasible .eqv. .false.) then
        write(2,'(a,f10.3,a)') 'JOINT PROBABILITY THAT TOTAL STOCK BIOMASS
EXCEEDS THRESHOLD:',(totBthresh/1.0e6),' THOUSAND MT'
        write(2,*) 'YEAR    Pr(B > Threshold Value)*Pr(Feasible)'
        do 62897 j=1,ntime
         write(2,'(1x,i4,12x,f5.3)') (baseyr+j-1),pjoint_totBthresh(j)
62897    continue
        write(2,*) ' '
         endif
    endif

    write(2,*) ' '
    write(2,*) 'RECRUITMENT UNITS ARE:',bootunit,' FISH'
    write(2,*) 'BIRTH       AVG'
    write(2,*) 'YEAR    RECRUITMENT      STD'
    do 5201 j=1,ntime
     write(2,'(1x,i4,4x,f10.3,4x,f10.3)') (baseyr+j-1),avgrecr(j),sdrecr(j)
5201   continue
    write(2,*) ' '
    write(2,*) 'PERCENTILES OF RECRUITMENT UNITS ARE:',bootunit,' FISH'
    write(2,*) 'BIRTH'
    write(2,*) 'YEAR      1%        5%        10%       25%       50%       75%       90%       95%
     99%'
    do 5251 j=1,ntime
     write(2,'(1x,i4,9(1x,f10.3))') (baseyr+j-1),(pctrecr(j,k),k=1,maxpct)
5251   continue

    if (indexflag .eqv. .true.) then
     write(2,*) ' '
     write(2,*) 'ESTIMATED SURVEY INDEX FOR AGE:',index_age
     write(2,*) 'YEAR    AVG INDEX (#)    STD'
     do 505 j=1,ntime
      write(2,'(1x,i4,1x,f10.3,1x,f10.3)') (baseyr+j-1),avgsvind(j),sdsvind(j)
505    continue
     write(2,*) ' '
     write(2,*) 'PERCENTILES OF SURVEY INDEX FOR AGE:',index_age
    write(2,*) 'YEAR      1%        5%        10%       25%              50%       75%       90%
95%       99%'
```

```
      do 506 j=1,ntime
        write(2,'(1x,i4,9(1x,f10.3))') (baseyr+j-1),(pctsvind(j,k),k=1,maxpct)
506     continue
      write(2,*) ' '
      write(2,*) 'ANNUAL PROBABILITY THAT SURVEY AGE INDEX
EXCEEDS:',crit_index
      write(2,*) 'YEAR    Pr(Index > Critical value) FOR FEASIBLE SIMULATIONS'
      do 5102 j=1,ntime
        write(2,'(1x,i4,9x,f5.3)') (baseyr+j-1),p_index(j)
5102    continue
      write(2,*) ' '
!c        REVISED JUL-2002
          if (allfeasible .eqv. .false.) then
        write(2,*) 'JOINT PROBABILITY THAT SURVEY AGE INDEX
EXCEEDS:',crit_index
        write(2,*) 'YEAR    Pr(Index > Critical value)*Pr(Feasible)'
        do 5107 j=1,ntime
          write(2,'(1x,i4,9x,f5.3)') (baseyr+j-1),pjoint_index(j)
5107      continue
        write(2,*) ' '
          endif
    endif

    if ((quotaflag .eqv. .false.) .or. (mixflag .eqv. .true.)) then
      write(2,*) ' '
      write(2,*) 'LANDINGS FOR F-BASED PROJECTIONS'
      write(2,*) 'YEAR    AVG LANDINGS (000 MT)   STD'
      do 530 j=1,ntime
        write(2,'(1x,i4,4x,f10.3,8x,f10.3)') (baseyr+j-1),(avgland(j)/1.0e6),(sdland(j)/1.0e6)
530     continue
    write(2,*) ' '
    write(2,*) 'PERCENTILES OF LANDINGS (000 MT)'
    write(2,*) 'YEAR      1%        5%        10%       25%       50%       75%       90%       95%
     99%'
    do 535 j=1,ntime
      write(2,'(1x,i4,9(1x,f10.3))') (baseyr+j-1),((pctland(j,k)/1.0e6),k=1,maxpct)
535   continue
    endif

    if (mcflag .eqv. .true.) then
!c   OUTPUT MARKET CATEGORY SUMMARIES
      write(2,*) ' '
      write(2,*) 'LANDINGS BY MARKET CATEGORY'
      do 4500 k=1,nmc
```

```
        write(2,*) ' '
        write(2,*) 'MARKET CATEGORY: ',k
        write(2,*) 'YEAR    AVG LANDINGS (000 MT)    STD'
        do 4530 j=1,ntime
          if (k .eq. 1) then
            write(2,'(1x,i4,4x,f10.3,8x,f10.3)')
(baseyr+j-1),(avgmc1_w(j)/1.0e6),(sdmc1_w(j)/1.0e6)
          else if (k .eq. 2) then
            write(2,'(1x,i4,4x,f10.3,8x,f10.3)')
(baseyr+j-1),(avgmc2_w(j)/1.0e6),(sdmc2_w(j)/1.0e6)
          else if (k .eq. 3) then
            write(2,'(1x,i4,4x,f10.3,8x,f10.3)')
(baseyr+j-1),(avgmc3_w(j)/1.0e6),(sdmc3_w(j)/1.0e6)
          endif
4530      continue
        write(2,*) ' '
        write(2,*) 'MARKET CATEGORY: ',k
        write(2,*) 'YEAR    AVG LANDINGS (000 FISH)    STD'
        do 4535 j=1,ntime
          if (k .eq. 1) then
            write(2,'(1x,i4,4x,f10.3,8x,d10.3)')
(baseyr+j-1),(avgmc1_n(j)/1.0e3),(sdmc1_n(j)/1.0e3)
          else if (k .eq. 2) then
            write(2,'(1x,i4,4x,f10.3,8x,d10.3)')
(baseyr+j-1),(avgmc2_n(j)/1.0e3),(sdmc2_n(j)/1.0e3)
          else if (k .eq. 3) then
            write(2,'(1x,i4,4x,f10.3,8x,d10.3)')
(baseyr+j-1),(avgmc3_n(j)/1.0e3),(sdmc3_n(j)/1.0e3)
          endif
4535      continue
        write(2,*) ' '
        write(2,*) 'PERCENTILES OF LANDINGS (000 MT)'
        write(2,*) 'MARKET CATEGORY: ',k
      write(2,*) 'YEAR      1%        5%        10%       25%       50%       75%       90%       95%
       99%'
        do 4635 j=1,ntime
          if (k .eq. 1) then
            write(2,'(1x,i4,9(1x,f10.3))') (baseyr+j-1),((pctmc1_w(j,kk)/1.0e6),kk=1,maxpct)
          else if (k .eq. 2) then
            write(2,'(1x,i4,9(1x,f10.3))') (baseyr+j-1),((pctmc2_w(j,kk)/1.0e6),kk=1,maxpct)
          else if (k .eq. 3) then
            write(2,'(1x,i4,9(1x,f10.3))') (baseyr+j-1),((pctmc3_w(j,kk)/1.0e6),kk=1,maxpct)
          endif
4635      continue
```

```
        write(2,*) ' '
        write(2,*) 'PERCENTILES OF LANDINGS (000 FISH)'
        write(2,*) 'MARKET CATEGORY: ',k
      write(2,*) 'YEAR      1%         5%         10%        25%              50%        75%        90%
95%        99%'
        do 4735 j=1,ntime
         if (k .eq. 1) then
          write(2,'(1x,i4,9(1x,f10.3))') (baseyr+j-1),((pctmc1_n(j,kk)/1.0e3),kk=1,maxpct)
         else if (k .eq. 2) then
          write(2,'(1x,i4,9(1x,f10.3))') (baseyr+j-1),((pctmc2_n(j,kk)/1.0e3),kk=1,maxpct)
         else if (k .eq. 3) then
          write(2,'(1x,i4,9(1x,f10.3))') (baseyr+j-1),((pctmc3_n(j,kk)/1.0e3),kk=1,maxpct)
         endif
4735     continue
4500    continue
      endif

      if (discflag .eqv. .true.) then
       write(2,*) ' '
       write(2,*) 'DISCARDS FOR F-BASED PROJECTIONS'
       write(2,*) 'YEAR  AVG DISCARDS (000 MT)    STD'
       do 540 j=1,ntime
        write(2,'(1x,i4,1x,f10.3,10x,f10.3)') (baseyr+j-1),(avgdisc(j)/1.0e6),(sddisc(j)/1.0e6)
540     continue
      write(2,*) ' '
      write(2,*) 'PERCENTILES OF DISCARDS (000 MT)'
      write(2,*) 'YEAR      1%         5%         10%        25%        50%        75%        90%        95%
       99%'
      do 545 j=1,ntime
       write(2,'(1x,i4,9(1x,f10.3))') (baseyr+j-1),((pctdisc(j,k)/1.0e6),k=1,maxpct)
545   continue
      endif

      if ((quotaflag .eqv. .true.) .or. (mixflag .eqv. .true.) .or.(ftarflag .eqv. .true.)) then
       write(2,*) ' '
       if (ftarflag .eqv. .false.) then
        write(2,*) 'REALIZED F SERIES FOR QUOTA-BASED PROJECTIONS'
       else
        write(2,*) 'REALIZED F SERIES FOR F-BASED PROJECTIONS'
        write(2,*) 'WITH A TARGET F OF: ',ftarget
       endif
       write(2,*) 'YEAR    AVG F     STD'
       do 550 j=1,ntime
        write(2,'(1x,i4,4x,f5.3,4x,f5.3)') (baseyr+j-1),avgf(j),sdf(j)
```

```
550     continue
        write(2,*) ' '
        write(2,*) 'PERCENTILES OF REALIZED F SERIES'
        write(2,*) 'YEAR    1%     5%     10%    25%    50%    75%    90%      95%     99%'
        do 555 j=1,ntime
          write(2,'(1x,i4,9(1x,f6.3))') (baseyr+j-1),(pctf(j,k),k=1,maxpct)
555     continue
      endif

!c        REVISED FEB-2002
      write(2,*) ' '
      if (sfaflag .eqv. .true.) then
        write(2,'(a,f10.3,a)') 'ANNUAL PROBABILITY FULLY-RECRUITED F EXCEEDS
THRESHOLD:',Fthresh
        write(2,*) 'YEAR    Pr(F > Threshold Value) FOR FEASIBLE SIMULATIONS'
        do 62999 j=1,ntime
          write(2,'(1x,i4,12x,f5.3)') (baseyr+j-1),p_Fthresh(j)
62999   continue
        write(2,*) ' '
!c        REVISED JUL-2002
          if (allfeasible .eqv. .false.) then
          write(2,'(a,f10.3,a)') 'JOINT PROBABILITY FULLY-RECRUITED F EXCEEDS
THRESHOLD:',Fthresh
          write(2,*) 'YEAR    Pr(F > Threshold Value)*Pr(Feasible)+(1.0-Pr(Feasible))'
          do 62997 j=1,ntime
            write(2,'(1x,i4,12x,f5.3)') (baseyr+j-1),pjoint_Fthresh(j)
62997     continue
          write(2,*) ' '
           endif
      endif

      close(2)

      print *,' '
      print *,'Projection analysis has been completed.'

99999 continue

      end
!c   END OF MAIN PROGRAM

!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!c          FUNCTION AND SUBROUTINE DECLARATIONS
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

```
      subroutine warmup(reps)
      integer*4 reps,idum
      real*8 ran2

      idum = -1
      do 10 i=1,reps
        z = ran2(idum)
10    continue
      return
      end
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine calc_catch(n,nage,maxage,f,pr,m,catch)
      integer*4 nage,maxage
      real*8 n(1:maxage),f,pr(1:maxage),m,catch(1:maxage)
      real*8 fage

!!c   APPLY THE CATCH EQUATION TO EACH AGE
      do 10 j=1,nage
        fage=f*pr(j)
        catch(j)=fage*n(j)*(1.-exp(-m-fage))/(m+fage)
10    continue
      return
      end
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine funcd(f,g_f,dg_f)
      use global_arrays
      real*8 sum,tmp1,tmp2,landings
      real*8 f,g_f,dg_f

!!c   COMPUTE FUNCTION VALUE (g_f) AND DERIVATIVE (dg_f) OF THE
!!c   FUNCTION g(f)=landings(f)-quota FOR A GIVEN VALUE OF f
!!c   COMPUTE CATCH AT AGE VECTOR GIVEN f
       call calc_catch(n,nage,maxage,f,pr,m,catch)
!!c   INITIALIZE landings AND sum
       landings=0.
       sum=0.

      if (discflag .eqv. .true.) then
!!c   COMPUTE LANDINGS BASED ON CATCH WITH DISCARDS
       do 10 j=1,nage
         landings=landings+catch(j)*(1.-discfrac(j))*wtland(j)
10     continue
!!c   COMPUTE DERIVATIVE dg_f AS sum WITH DISCARDS
       do 15 j=1,nage
```

```
        tmp1=exp(-m-pr(j)*f)

tmp2=(1-discfrac(j))*wtland(j)*pr(j)*n(j)*(m+tmp1*(m*pr(j)*f-m+(pr(j)*f)**2.0))/(m+pr(j)*f)
**2.0
        sum=sum+tmp2
15      continue
      else
!!c   COMPUTE LANDINGS BASED ON CATCH WITH NO DISCARDS
       do 20 j=1,nage
         landings=landings+catch(j)*wtland(j)
20       continue
!!c   COMPUTE DERIVATIVE dg_f AS sum WITH NO DISCARDS
       do 25 j=1,nage
         tmp1=exp(-m-pr(j)*f)
         tmp2=wtland(j)*pr(j)*n(j)*(m+tmp1*(m*pr(j)*f-m+(pr(j)*f)**2.0))/(m+pr(j)*f)**2.0
         sum=sum+tmp2
25       continue
      endif
!!c   COMPUTE g_f
      g_f=landings-quota
!!c   COMPUTE dg_f
      dg_f=sum

      return
      end
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      real function calc_ssb(n,nage,maxage,pr,m,f,zproj,fm,wt)
      integer*4 nage,maxage
      real*8 n(1:maxage),pr(1:maxage),m,f,zproj,fm(1:maxage)
      real*8 wt(1:maxage),sum,fage

      sum=0.0
      do 10 j=1,nage
        fage=pr(j)*f
        sum=sum+exp(-zproj*(m+fage))*fm(j)*wt(j)*n(j)
10    continue
      calc_ssb=sum
      return
      end
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!c        REVISED 7/1/99
      real function calc_meanB(n,lowerage,upperage,maxage,pr,m,f,wtland)
      integer*4 lowerage,upperage,maxage
      real*8 n(1:maxage),pr(1:maxage),m,f
```

```
      real*8 wtland(1:maxage),sum,fage

      sum=0.0
      do 10 j=lowerage,upperage
        fage=pr(j)*f
        sum=sum+wtland(j)*n(j)*(1.0-exp(-m-fage))/(m+fage)
10    continue
      calc_meanB=sum
      return
      end
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!c        REVISED FEB-2002
      real function calc_totB(n,maxage,wt)
      integer*4 maxage
      real*8 n(1:maxage)
      real*8 wt(1:maxage),sum

      sum=0.0
      do 10 j=1,maxage
        sum=sum+wt(j)*n(j)
10    continue
      calc_totB=sum
      return
      end
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!c        REVISED 7/8/99
      real function calc_FmeanB(n,lowerage,upperage,maxage,pr,m,f,wtland)
      integer*4 lowerage,upperage,maxage
      real*8 n(1:maxage),pr(1:maxage),m,f
      real*8 wtland(1:maxage),tmp,sum1,sum2,fage

      sum1=0.0
      sum2=0.0
      do 10 j=lowerage,upperage
        fage=pr(j)*f
        tmp=wtland(j)*n(j)*(1.0-exp(-m-fage))/(m+fage)
        sum1=sum1+fage*tmp
        sum2=sum2+tmp
10    continue
      calc_FmeanB=sum1/sum2
      return
      end
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine calc_next_n(n,nage,pr,m,f,reclevel,next_n)
```

```
      parameter (maxage=50)
      integer*4 nage
      real*8 n(1:maxage),pr(1:maxage),m,f,reclevel,next_n(1:maxage)
      real*8 fage(1:maxage)

!!c   COMPUTE F-AT-AGE
      do 10 j=1,nage
        fage(j)=f*pr(j)
10    continue

!!c   SET RECRUITMENT
      next_n(1)=reclevel

!!c   SET next_n FOR AGES 2 TO nage-1
      do 20 j=2,(nage-1)
        next_n(j)=n(j-1)*exp(-m-fage(j-1))
20    continue

!!c   SET next_n FOR THE PLUS-GROUP
      next_n(nage)=n(nage)*exp(-m-fage(nage))+n(nage-1)*exp(-m-fage(nage-1))
      return
      end
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine simavg(simdat,nboot,nsim,ntime,quotaflag,mixflag,nfeasible,avgdat)
      use global_arrays
      integer*4 nboot,nsim,ntime,nfeasible
      logical quotaflag,mixflag
      real*8 simdat(1:maxboot,1:maxsim,1:maxtime),avgdat(1:maxtime)
      logical dum

!!c   SUM SIMULATION DATA
      if ((quotaflag .eqv. .true.) .or. (mixflag .eqv. .true.)) then
        do 30 j=1,ntime
          avgdat(j)=0.0
          do 20 jj=1,nboot
            do 10 jjj=1,nsim
                dum=feasible(jj,jjj)
              if (dum .eqv. .true.) then
                  avgdat(j)=avgdat(j)+simdat(jj,jjj,j)
              endif
10            continue
20          continue
30        continue
      else
```

```
      do 40 j=1,ntime
        avgdat(j)=0.0
        do 50 jj=1,nboot
          do 60 jjj=1,nsim
            avgdat(j)=avgdat(j)+simdat(jj,jjj,j)
60          continue
50        continue
40      continue
      endif

!c    COMPUTE AVERAGES
      do 100 j=1,ntime
        avgdat(j)=avgdat(j)/nfeasible
100   continue
      return
      end
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine simsd(simdat,avgdat,nboot,nsim,ntime,quotaflag,mixflag,nfeasible,sddat)
      use global_arrays
      integer*4 nboot,nsim,ntime,nfeasible
      logical quotaflag,mixflag
      real*8 simdat(1:maxboot,1:maxsim,1:maxtime),avgdat(1:maxtime)
      real*8 sddat(1:maxtime)
      logical dum

      if ((quotaflag .eqv. .true.) .or. (mixflag .eqv. .true.)) then
        do 10 j=1,ntime
          sddat(j)=0.
          do 20 jj=1,nboot
            do 30 jjj=1,nsim
                dum=feasible(jj,jjj)
              if (dum .eqv. .true.) then
                sddat(j)=sddat(j)+(simdat(jj,jjj,j)-avgdat(j))**2.
              endif
30          continue
20        continue
10      continue
      else
        do 40 j=1,ntime
          sddat(j)=0.
          do 50 jj=1,nboot
            do 60 jjj=1,nsim
              sddat(j)=sddat(j)+(simdat(jj,jjj,j)-avgdat(j))**2.
60          continue
```

```
50        continue
40      continue
      endif

!c   COMPUTE SAMPLE STANDARD DEVIATION
      do 100 j=1,ntime
        sddat(j)=sddat(j)/(nfeasible-1.)
        sddat(j)=sddat(j)**0.5
100   continue

      return
      end
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE hpsort(n,ra)
      integer*4 n
      REAL*8 ra(n)
      integer*4 i,ir,j,l
      REAL*8 rra

      if (n.lt.2) return
      l=n/2+1
      ir=n
10    continue
        if(l.gt.1)then
          l=l-1
          rra=ra(l)
        else
          rra=ra(ir)
          ra(ir)=ra(1)
          ir=ir-1
          if(ir.eq.1)then
            ra(1)=rra
            return
          endif
        endif
        i=l
        j=l+l
20      if(j.le.ir)then
          if(j.lt.ir)then
            if(ra(j).lt.ra(j+1))j=j+1
          endif
          if(rra.lt.ra(j))then
            ra(i)=ra(j)
            i=j
```

```
          j=j+j
         else
          j=ir+1
         endif
        goto 20
        endif
        ra(i)=rra
      goto 10
      END
!C  (C) Copr. 1986-92 Numerical Recipes Software *1."21$:)!+.
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine summarize(simdat,avgdat,sddat,pctdat,quotaflag,mixflag)
      use global_arrays
!!c  SUMMARIZE THE OUTPUT STATISTICALLLY: MEANS, STD, PERCENTILES

      real*8 simdat(1:maxboot,1:maxsim,1:maxtime), avgdat(1:maxtime)
      real*8 sddat(1:maxtime),sortarray(1:maxlen)
!c        REVISED 6/5/02
      real*8 pctdat(1:maxtime,1:maxpct)
      real*8 pctvalue(1:maxpct)
      integer*4 j,jj,time,index,iflag
      integer*4 k,nsim,ntime,nboot,pctptr
      integer*4 nfeasible
      logical quotaflag,mixflag

!c        REVISED 6/5/02
      common /params/ nfeasible,iflag,nboot,nsim,ntime
      common /pcxval/ pctvalue

!!c   COMPUTE TIME-AVERAGES FOR simdat
      call simavg(simdat,nboot,nsim,ntime,quotaflag,mixflag,nfeasible,avgdat)

!!c   COMPUTE STANDARD DEVIATION OF simdat BY TIME PERIOD
      call simsd(simdat,avgdat,nboot,nsim,ntime,quotaflag,mixflag,nfeasible,sddat)

!!c   COMPUTE PERCENTILES OF simdat DISTRIBUTION BY TIME PERIOD
       do 750 time=1,ntime
!!c   INITIALIZE THE ARRAY
         index=1
         do 755 j=1,maxlen
           sortarray(j)=-1.
755       continue
         do 760 j=1,nboot
           do 770 jj=1,nsim
```

```
          if (feasible(j,jj) .eqv..true.) then
            sortarray(index)=simdat(j,jj,time)
            index=index+1
          endif
770       continue
760      continue

!!c   SORT THE ARRAY
        call hpsort(maxlen,sortarray)

!!c   COMPUTE PERCENTILES (1,5,10,25,50,75,90,95,99)

         do k = 1,maxpct
           pctptr=nint(nfeasible*pctvalue(k))+iflag
           pctdat(time,k)=sortarray(pctptr)
        end do

750    continue
    return
    end
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
    FUNCTION ran2(idum)
    integer*4 idum,IM1,IM2,IMM1,IA1,IA2,IQ1,IQ2,IR1,IR2,NTAB,NDIV
    REAL*8 ran2,AM,EPS,RNMX
    PARAMETER (IM1=2147483563,IM2=2147483399,AM=1./IM1,IMM1=IM1-1)
    Parameter (IA1=40014,IA2=40692,IQ1=53668,IQ2=52774,IR1=12211)
    parameter (IR2=3791,NTAB=32,NDIV=1+IMM1/NTAB,EPS=1.2e-7,RNMX=1.-EPS)
    integer*4 idum2,j,k,iv(NTAB),iy

    SAVE iv,iy,idum2
    DATA idum2/123456789/, iv/NTAB*0/, iy/0/

    if (idum.le.0) then
      idum=max(-idum,1)
      idum2=idum
      do 11 j=NTAB+8,1,-1
        k=idum/IQ1
        idum=IA1*(idum-k*IQ1)-k*IR1
        if (idum.lt.0) idum=idum+IM1
        if (j.le.NTAB) iv(j)=idum
11      continue
      iy=iv(1)
    endif
    k=idum/IQ1
```

```
      idum=IA1*(idum-k*IQ1)-k*IR1
      if (idum.lt.0) idum=idum+IM1
      k=idum2/IQ2
      idum2=IA2*(idum2-k*IQ2)-k*IR2
      if (idum2.lt.0) idum2=idum2+IM2
      j=1+iy/NDIV
      iy=iv(j)-idum2
      iv(j)=idum
      if(iy.lt.1)iy=iy+IMM1
      ran2=min(AM*iy,RNMX)
      return
      END
!C  (C) Copr. 1986-92 Numerical Recipes Software *1."21$:)!+.
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      FUNCTION gasdev(idum)
      integer*4 idum
      REAL*8 gasdev
!CU    USES ran2
      integer*4 iset
      REAL*8 fac,gset,rsq,v1,v2,ran2

      SAVE iset,gset
      DATA iset/0/

      if (iset.eq.0) then
1       v1=2.*ran2(idum)-1.
        v2=2.*ran2(idum)-1.
        rsq=v1**2+v2**2
        if(rsq.ge.1..or.rsq.eq.0.)goto 1
        fac=sqrt(-2.*log(rsq)/rsq)
        gset=v1*fac
        gasdev=v2*fac
        iset=1
      else
        gasdev=gset
        iset=0
      endif
      return
      END
!C  (C) Copr. 1986-92 Numerical Recipes Software *1."21$:)!+.
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      FUNCTION rtsafe(x1,x2,xacc)
      integer*4 MAXIT
      REAL*8 rtsafe,x1,x2,xacc
```

```
PARAMETER (MAXIT=100)
integer*4 j
REAL*8 df,dx,dxold,f,fh,fl,temp,xh,xl

call funcd(x1,fl,df)
call funcd(x2,fh,df)
if((fl.gt.0..and.fh.gt.0.).or.(fl.lt.0..and.fh.lt.0.))pause 'root must be bracketed in rtsafe'
if(fl.eq.0.)then
  rtsafe=x1
  return
else if(fh.eq.0.)then
  rtsafe=x2
  return
else if(fl.lt.0.)then
  xl=x1
  xh=x2
else
  xh=x1
  xl=x2
endif
rtsafe=.5*(x1+x2)
dxold=abs(x2-x1)
dx=dxold
call funcd(rtsafe,f,df)
do 11 j=1,MAXIT
  if(((rtsafe-xh)*df-f)*((rtsafe-xl)*df-f).ge.0..or. abs(2.*f).gt.abs(dxold*df) ) then
    dxold=dx
    dx=0.5*(xh-xl)
    rtsafe=xl+dx
    if(xl.eq.rtsafe)return
  else
    dxold=dx
    dx=f/df
    temp=rtsafe
    rtsafe=rtsafe-dx
    if(temp.eq.rtsafe)return
  endif
  if(abs(dx).lt.xacc) return
  call funcd(rtsafe,f,df)
  if(f.lt.0.) then
    xl=rtsafe
  else
    xh=rtsafe
  endif
```

```
11    continue
      pause 'rtsafe exceeding maximum iterations'
      return
      END
!C  (C) Copr. 1986-92 Numerical Recipes Software *1."21$:)!+.
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine allocation
!c        REVISED FEB-2002
      use global_arrays

      allocate(feasible(1:maxboot,1:maxsim))
      allocate( boot_n(1:maxboot,1:maxage))
      allocate( boot_f(1:maxboot))
      allocate( simssb(1:maxboot,1:maxsim,1:maxtime))
!c        REVISED 7/1/99
      allocate( simmeanB(1:maxboot,1:maxsim,1:maxtime))
      allocate( simtotB(1:maxboot,1:maxsim,1:maxtime))
      allocate( simland(1:maxboot,1:maxsim,1:maxtime))
      allocate( simdisc(1:maxboot,1:maxsim,1:maxtime))
      allocate( simf(1:maxboot,1:maxsim,1:maxtime))
!c        REVISED 7/8/99
      allocate( simFmeanB(1:maxboot,1:maxsim,1:maxtime))
      allocate( simrecr(1:maxboot,1:maxsim,1:maxtime))
      allocate( simsvind(1:maxboot,1:maxsim,1:maxtime))
      end
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine mcAlloc
      use global_arrays

      allocate( simmc1_w(1:maxboot,1:maxsim,1:maxtime))
      allocate( simmc1_n(1:maxboot,1:maxsim,1:maxtime))
      allocate( simmc2_w(1:maxboot,1:maxsim,1:maxtime))
      allocate( simmc2_n(1:maxboot,1:maxsim,1:maxtime))
      allocate( simmc3_w(1:maxboot,1:maxsim,1:maxtime))
      allocate( simmc3_n(1:maxboot,1:maxsim,1:maxtime))
      end
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine OtherAlloc
      use global_arrays
      allocate( mixyr(1:maxtime))
      allocate( var_pr(1:maxtime,1:maxage),maxcatch(1:maxage))
      allocate( var_discfrac(1:maxtime,1:maxage))
      allocate( var_zproj(1:maxtime))
      allocate(fseries(1:maxtime),qseries(1:maxtime))
```

```
      allocate( avgssb(1:maxtime),avgland(1:maxtime),avgsvind(1:maxtime))
      allocate( avgdisc(1:maxtime),avgf(1:maxtime),avgrecr(1:maxtime))
!c       REVISED 7/1/99
      allocate( avgmeanB(1:maxtime),sdmeanB(1:maxtime))
      allocate( avgtotB(1:maxtime),sdtotB(1:maxtime))
      allocate( sdssb(1:maxtime),sdland(1:maxtime),sdsvind(1:maxtime))
      allocate( sddisc(1:maxtime),sdf(1:maxtime),sdrecr(1:maxtime))
      allocate( pctssb(1:maxtime,1:maxpct),pctdisc(1:maxtime,1:maxpct))
      allocate( pctland(1:maxtime,1:maxpct),pctf(1:maxtime,1:maxpct))
      allocate( pctrecr(1:maxtime,1:maxpct),pctsvind(1:maxtime,1:maxpct))
!c       REVISED 7/1/99
      allocate( pctmeanB(1:maxtime,1:maxpct))
      allocate( pcttotB(1:maxtime,1:maxpct))
      allocate( crit_count(1:maxtime),p_index(1:maxtime))
      allocate( ssb_count(1:maxtime),p_ssbthresh(1:maxtime))
!c       REVISED 7/8/99
      allocate( meanB_count(1:maxtime),p_meanBthresh(1:maxtime))
      allocate( totB_count(1:maxtime),p_totBthresh(1:maxtime))
      allocate( FmeanB_count(1:maxtime),p_FmeanBthresh(1:maxtime))
      allocate( F_count(1:maxtime),p_Fthresh(1:maxtime))
      allocate( avgFmeanB(1:maxtime),sdFmeanB(1:maxtime),pctFmeanB(1:maxtime,1:maxpct))
!c       REVISED JUL-2002
      allocate( pjoint_index(1:maxtime))
      allocate( pjoint_ssbthresh(1:maxtime))
      allocate( pjoint_meanBthresh(1:maxtime))
      allocate( pjoint_totBthresh(1:maxtime))
      allocate( pjoint_FmeanBthresh(1:maxtime))
      allocate( pjoint_Fthresh(1:maxtime))
      allocate( market(1:maxmc,1:maxage))
      allocate( avgmc1_w(1:maxtime))
      allocate( avgmc1_n(1:maxtime))
      allocate( avgmc2_w(1:maxtime))
      allocate( avgmc2_n(1:maxtime))
      allocate( avgmc3_w(1:maxtime))
      allocate( avgmc3_n(1:maxtime))
      allocate( sdmc1_w(1:maxtime))
      allocate( sdmc1_n(1:maxtime))
      allocate( sdmc2_w(1:maxtime))
      allocate( sdmc2_n(1:maxtime))
      allocate( sdmc3_w(1:maxtime))
      allocate( sdmc3_n(1:maxtime))
      allocate( pctmc1_w(1:maxtime,1:maxpct))
      allocate( pctmc1_n(1:maxtime,1:maxpct))
      allocate( pctmc2_w(1:maxtime,1:maxpct))
```

```
      allocate( pctmc2_n(1:maxtime,1:maxpct))
      allocate( pctmc3_w(1:maxtime,1:maxpct))
      allocate( pctmc3_n(1:maxtime,1:maxpct))
      allocate(obsrec3(1:maxtime,1:maxobsrec))
      allocate(resid(0:maxtime))
      end
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```